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 compile() . . . 13
5.1.3.2 compilePreprocess() . . . 14
5.1.3.3 dump() . . . 14
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 compile() . . . 17
5.2.3.2 compilePreprocess() . . . 17
5.2.3.3 dump() . . . 18
5.3 Tang::AstNodeBinary Class Reference . . . 18
5.3.1 Detailed Description . . . 19
5.3.2 Member Enumeration Documentation . . . 19
5.3.2.1 Operation . . . 19
5.3.3 Constructor & Destructor Documentation . . . 20
5.3.3.1 AstNodeBinary() . . . 20
5.3.4 Member Function Documentation . . . 20
5.3.4.1 compile() . . . 20
5.3.4.2 compilePreprocess() . . . 21
5.3.4.3 dump() . . . 21
5.4 Tang::AstNodeBlock Class Reference . . . 22
5.4.1 Detailed Description . . . 23
5.4.2 Constructor & Destructor Documentation . . . 23

5.4.2.1 AstNodeBlock() . . . 23
5.4.3 Member Function Documentation . . . 23
5.4.3.1 compile() . . . 23
5.4.3.2 compilePreprocess() . . . 24
5.4.3.3 dump() . . . 24
5.5 Tang::AstNodeBoolean Class Reference . . . 25
5.5.1 Detailed Description . . . 25
5.5.2 Constructor & Destructor Documentation . . . 26
5.5.2.1 AstNodeBoolean() . . . 26
5.5.3 Member Function Documentation . . . 26
5.5.3.1 compile() . . . 26
5.5.3.2 compilePreprocess() . . . 27
5.5.3.3 dump() . . . 27
5.6 Tang::AstNodeCast Class Reference . . . 27
5.6.1 Detailed Description . . . 28
5.6.2 Member Enumeration Documentation . . . 29
5.6.2.1 Type . . . 29
5.6.3 Constructor & Destructor Documentation . . . 29
5.6.3.1 AstNodeCast() . . . 29
5.6.4 Member Function Documentation . . . 29
5.6.4.1 compile() . . . 29
5.6.4.2 compilePreprocess() . . . 30
5.6.4.3 dump() . . . 30
5.7 Tang::AstNodeDoWhile Class Reference . . . 31
5.7.1 Detailed Description . . . 32
5.7.2 Constructor & Destructor Documentation . . . 32
5.7.2.1 AstNodeDoWhile() . . . 32
5.7.3 Member Function Documentation . . . 32
5.7.3.1 compile() . . . 32
5.7.3.2 compilePreprocess() . . . 33
5.7.3.3 dump() . . . 33
5.8 Tang::AstNodeFloat Class Reference . . . 34
5.8.1 Detailed Description . . . 35
5.8.2 Constructor & Destructor Documentation . . . 35
5.8.2.1 AstNodeFloat() . . . 35
5.8.3 Member Function Documentation . . . 35
5.8.3.1 compile() . . . 35
5.8.3.2 compilePreprocess() . . . 36
5.8.3.3 dump() . . . 36
5.9 Tang::AstNodeFor Class Reference . . . 37
5.9.1 Detailed Description . . . 38
5.9.2 Constructor & Destructor Documentation . . . 38

5.9.2.1 AstNodeFor() . . . 38
5.9.3 Member Function Documentation . . . 38
5.9.3.1 compile() . . . 38
5.9.3.2 compilePreprocess() . . . 39
5.9.3.3 dump() . . . 39
5.10 Tang::AstNodeFunctionCall Class Reference . . . 40
5.10.1 Detailed Description . . . 41
5.10.2 Constructor & Destructor Documentation . . . 41
5.10.2.1 AstNodeFunctionCall() . . . 41
5.10.3 Member Function Documentation . . . 41
5.10.3.1 compile() . . . 41
5.10.3.2 compilePreprocess() . . . 42
5.10.3.3 dump() . . . 42
5.11 Tang::AstNodeFunctionDeclaration Class Reference . . . 43
5.11.1 Detailed Description . . . 44
5.11.2 Constructor & Destructor Documentation . . . 44
5.11.2.1 AstNodeFunctionDeclaration() . . . 44
5.11.3 Member Function Documentation . . . 44
5.11.3.1 compile() . . . 44
5.11.3.2 compilePreprocess() . . . 45
5.11.3.3 dump() . . . 45
5.12 Tang::AstNodeIdentifier Class Reference . . . 46
5.12.1 Detailed Description . . . 47
5.12.2 Constructor & Destructor Documentation . . . 47
5.12.2.1 AstNodeIdentifier() . . . 47
5.12.3 Member Function Documentation . . . 48
5.12.3.1 compile() . . . 48
5.12.3.2 compilePreprocess() . . . 48
5.12.3.3 dump() . . . 49
5.13 Tang::AstNodeIfElse Class Reference . . . 49
5.13.1 Detailed Description . . . 50
5.13.2 Constructor & Destructor Documentation . . . 51
5.13.2.1 AstNodeIfElse() [1/2] . . . 51
5.13.2.2 AstNodeIfElse() [2/2] . . . 51
5.13.3 Member Function Documentation . . . 51
5.13.3.1 compile() . . . 51
5.13.3.2 compilePreprocess() . . . 52
5.13.3.3 dump() . . . 52
5.14 Tang::AstNodeInteger Class Reference . . . 53
5.14.1 Detailed Description . . . 54
5.14.2 Constructor & Destructor Documentation . . . 54
5.14.2.1 AstNodeInteger() . . . 54

5.14.3 Member Function Documentation . . . 54
5.14.3.1 compile() . . . 54
5.14.3.2 compilePreprocess() . . . 55
5.14.3.3 dump() . . . 55
5.15 Tang::AstNodePrint Class Reference . . . 56
5.15.1 Detailed Description . . . 57
5.15.2 Member Enumeration Documentation . . . 57
5.15.2.1 Type . . . 57
5.15.3 Constructor & Destructor Documentation . . . 57
5.15.3.1 AstNodePrint() . . . 57
5.15.4 Member Function Documentation . . . 58
5.15.4.1 compile() . . . 58
5.15.4.2 compilePreprocess() . . . 58
5.15.4.3 dump() . . . 58
5.16 Tang::AstNodeReturn Class Reference . . . 59
5.16.1 Detailed Description . . . 60
5.16.2 Constructor & Destructor Documentation . . . 60
5.16.2.1 AstNodeReturn() . . . 60
5.16.3 Member Function Documentation . . . 60
5.16.3.1 compile() . . . 60
5.16.3.2 compilePreprocess() . . . 61
5.16.3.3 dump() . . . 61
5.17 Tang::AstNodeString Class Reference . . . 62
5.17.1 Detailed Description . . . 63
5.17.2 Constructor & Destructor Documentation . . . 63
5.17.2.1 AstNodeString() . . . 63
5.17.3 Member Function Documentation . . . 63
5.17.3.1 compile() . . . 63
5.17.3.2 compileLiteral() . . . 64
5.17.3.3 compilePreprocess() . . . 65
5.17.3.4 dump() . . . 65
5.18 Tang::AstNodeTernary Class Reference . . . 66
5.18.1 Detailed Description . . . 67
5.18.2 Constructor & Destructor Documentation . . . 67
5.18.2.1 AstNodeTernary() . . . 67
5.18.3 Member Function Documentation . . . 67
5.18.3.1 compile() . . . 67
5.18.3.2 compilePreprocess() . . . 68
5.18.3.3 dump() . . . 68
5.19 Tang::AstNodeUnary Class Reference . . . 69
5.19.1 Detailed Description . . . 70
5.19.2 Member Enumeration Documentation . . . 70

5.19.2.1 Operator . . . 70
5.19.3 Constructor & Destructor Documentation . . . 70
5.19.3.1 AstNodeUnary() . . . 70
5.19.4 Member Function Documentation . . . 71
5.19.4.1 compile() . . . 71
5.19.4.2 compilePreprocess() . . . 71
5.19.4.3 dump() . . . 71
5.20 Tang::AstNodeWhile Class Reference . . . 72
5.20.1 Detailed Description . . . 73
5.20.2 Constructor & Destructor Documentation . . . 73
5.20.2.1 AstNodeWhile() . . . 73
5.20.3 Member Function Documentation . . . 73
5.20.3.1 compile() . . . 73
5.20.3.2 compilePreprocess() . . . 74
5.20.3.3 dump() . . . 74
5.21 Tang::ComputedExpression Class Reference . . . 75
5.21.1 Detailed Description . . . 76
5.21.2 Member Function Documentation . . . 76
5.21.2.1 __add() . . . 77
5.21.2.2 __boolean() . . . 77
5.21.2.3 __divide() . . . 77
5.21.2.4 __equal() . . . 78
5.21.2.5 __float() . . . 78
5.21.2.6 __integer() . . . 78
5.21.2.7 __lessThan() . . . 78
5.21.2.8 __modulo() . . . 79
5.21.2.9 __multiply() . . . 79
5.21.2.10 __negative() . . . 80
5.21.2.11 __not() . . . 80
5.21.2.12 __string() . . . 80
5.21.2.13 __subtract() . . . 80
5.21.2.14 dump() . . . 81
5.21.2.15 is_equal() [1/6] . . . 81
5.21.2.16 is_equal() [2/6] . . . 82
5.21.2.17 is_equal() [3/6] . . . 82
5.21.2.18 is_equal() [4/6] . . . 82
5.21.2.19 is_equal() [5/6] . . . 83
5.21.2.20 is_equal() [6/6] . . . 83
5.21.2.21 makeCopy() . . . 83
5.22 Tang::ComputedExpressionBoolean Class Reference . . . 84
5.22.1 Detailed Description . . . 85
5.22.2 Constructor & Destructor Documentation . . . 86

5.22.2.1 ComputedExpressionBoolean() . . . 86
5.22.3 Member Function Documentation . . . 86
5.22.3.1 __add() . . . 86
5.22.3.2 __boolean() . . . 86
5.22.3.3 __divide() . . . 87
5.22.3.4 __equal() . . . 87
5.22.3.5 __float() . . . 87
5.22.3.6 __integer() . . . 88
5.22.3.7 __lessThan() . . . 88
5.22.3.8 __modulo() . . . 88
5.22.3.9 __multiply() . . . 89
5.22.3.10 __negative() . . . 89
5.22.3.11 __not() . . . 89
5.22.3.12 __string() . . . 90
5.22.3.13 __subtract() . . . 90
5.22.3.14 dump() . . . 90
5.22.3.15 is_equal() [1/6] . . . 90
5.22.3.16 is_equal() [2/6] . . . 91
5.22.3.17 is_equal() [3/6] . . . 91
5.22.3.18 is_equal() [4/6] . . . 92
5.22.3.19 is_equal() [5/6] . . . 92
5.22.3.20 is_equal() [6/6] . . . 92
5.22.3.21 makeCopy() . . . 93
5.23 Tang::ComputedExpressionCompiledFunction Class Reference . . . 93
5.23.1 Detailed Description . . . 95
5.23.2 Constructor & Destructor Documentation . . . 95
5.23.2.1 ComputedExpressionCompiledFunction() . . . 95
5.23.3 Member Function Documentation . . . 95
5.23.3.1 __add() . . . 95
5.23.3.2 __boolean() . . . 96
5.23.3.3 __divide() . . . 96
5.23.3.4 __equal() . . . 97
5.23.3.5 __float() . . . 97
5.23.3.6 __integer() . . . 97
5.23.3.7 __lessThan() . . . 97
5.23.3.8 __modulo() . . . 98
5.23.3.9 __multiply() . . . 98
5.23.3.10 __negative() . . . 99
5.23.3.11 __not() . . . 99
5.23.3.12 __string() . . . 99
5.23.3.13 __subtract() . . . 99
5.23.3.14 dump() . . . 100

5.23.3.15 is_equal() [1/6] . . . 100
5.23.3.16 is_equal() [2/6] . . . 101
5.23.3.17 is_equal() [3/6] . . . 102
5.23.3.18 is_equal() [4/6] . . . 102
5.23.3.19 is_equal() [5/6] . . . 103
5.23.3.20 is_equal() [6/6] . . . 103
5.23.3.21 makeCopy() . . . 103
5.24 Tang::ComputedExpressionError Class Reference . . . 104
5.24.1 Detailed Description . . . 105
5.24.2 Constructor & Destructor Documentation . . . 105
5.24.2.1 ComputedExpressionError() . . . 105
5.24.3 Member Function Documentation . . . 106
5.24.3.1 __add() . . . 106
5.24.3.2 __boolean() . . . 106
5.24.3.3 __divide() . . . 106
5.24.3.4 __equal() . . . 107
5.24.3.5 __float() . . . 107
5.24.3.6 __integer() . . . 108
5.24.3.7 __lessThan() . . . 108
5.24.3.8 __modulo() . . . 108
5.24.3.9 __multiply() . . . 109
5.24.3.10 __negative() . . . 109
5.24.3.11 __not() . . . 109
5.24.3.12 __string() . . . 110
5.24.3.13 __subtract() . . . 110
5.24.3.14 dump() . . . 110
5.24.3.15 is_equal() [1/6] . . . 110
5.24.3.16 is_equal() [2/6] . . . 111
5.24.3.17 is_equal() [3/6] . . . 111
5.24.3.18 is_equal() [4/6] . . . 112
5.24.3.19 is_equal() [5/6] . . . 112
5.24.3.20 is_equal() [6/6] . . . 112
5.24.3.21 makeCopy() . . . 113
5.25 Tang::ComputedExpressionFloat Class Reference . . . 113
5.25.1 Detailed Description . . . 115
5.25.2 Constructor & Destructor Documentation . . . 115
5.25.2.1 ComputedExpressionFloat() . . . 115
5.25.3 Member Function Documentation . . . 115
5.25.3.1 __add() . . . 115
5.25.3.2 __boolean() . . . 116
5.25.3.3 __divide() . . . 116
5.25.3.4 __equal() . . . 116

5.25.3.5 __float() . . . 117
5.25.3.6 __integer() . . . 117
5.25.3.7 __lessThan() . . . 117
5.25.3.8 __modulo() . . . 118
5.25.3.9 __multiply() . . . 118
5.25.3.10 __negative() . . . 119
5.25.3.11 __not() . . . 119
5.25.3.12 __string() . . . 119
5.25.3.13 __subtract() . . . 119
5.25.3.14 dump() . . . 120
5.25.3.15 is_equal() [1/6] . . . 120
5.25.3.16 is_equal() [2/6] . . . 120
5.25.3.17 is_equal() [3/6] . . . 121
5.25.3.18 is_equal() [4/6] . . . 121
5.25.3.19 is_equal() [5/6] . . . 122
5.25.3.20 is_equal() [6/6] . . . 122
5.25.3.21 makeCopy() . . . 122
5.26 Tang::ComputedExpressionInteger Class Reference . . . 123
5.26.1 Detailed Description . . . 124
5.26.2 Constructor & Destructor Documentation . . . 124
5.26.2.1 ComputedExpressionInteger() . . . 124
5.26.3 Member Function Documentation . . . 125
5.26.3.1 __add() . . . 125
5.26.3.2 __boolean() . . . 125
5.26.3.3 __divide() . . . 125
5.26.3.4 __equal() . . . 126
5.26.3.5 __float() . . . 126
5.26.3.6 __integer() . . . 127
5.26.3.7 __lessThan() . . . 127
5.26.3.8 __modulo() . . . 127
5.26.3.9 __multiply() . . . 128
5.26.3.10 __negative() . . . 128
5.26.3.11 __not() . . . 128
5.26.3.12 __string() . . . 129
5.26.3.13 __subtract() . . . 129
5.26.3.14 dump() . . . 129
5.26.3.15 is_equal() [1/6] . . . 130
5.26.3.16 is_equal() [2/6] . . . 130
5.26.3.17 is_equal() [3/6] . . . 130
5.26.3.18 is_equal() [4/6] . . . 131
5.26.3.19 is_equal() [5/6] . . . 131
5.26.3.20 is_equal() [6/6] . . . 132

5.26.3.21 makeCopy() . . . 132
5.27 Tang::ComputedExpressionString Class Reference . . . 132
5.27.1 Detailed Description . . . 134
5.27.2 Constructor & Destructor Documentation . . . 134
5.27.2.1 ComputedExpressionString() . . . 134
5.27.3 Member Function Documentation . . . 135
5.27.3.1 __add() . . . 135
5.27.3.2 __boolean() . . . 135
5.27.3.3 __divide() . . . 135
5.27.3.4 __equal() . . . 136
5.27.3.5 __float() . . . 136
5.27.3.6 __integer() . . . 136
5.27.3.7 __lessThan() . . . 137
5.27.3.8 __modulo() . . . 137
5.27.3.9 __multiply() . . . 137
5.27.3.10 __negative() . . . 138
5.27.3.11 __not() . . . 138
5.27.3.12 __string() . . . 138
5.27.3.13 __subtract() . . . 138
5.27.3.14 dump() . . . 139
5.27.3.15 is_equal() [1/6] . . . 139
5.27.3.16 is_equal() [2/6] . . . 139
5.27.3.17 is_equal() [3/6] . . . 140
5.27.3.18 is_equal() [4/6] . . . 140
5.27.3.19 is_equal() [5/6] . . . 141
5.27.3.20 is_equal() [6/6] . . . 141
5.27.3.21 makeCopy() . . . 141
5.28 Tang::Error Class Reference . . . 142
5.28.1 Detailed Description . . . 143
5.28.2 Constructor & Destructor Documentation . . . 143
5.28.2.1 Error() [1/2] . . . 143
5.28.2.2 Error() [2/2] . . . 143
5.28.3 Friends And Related Function Documentation . . . 143
5.28.3.1 operator<< . . . 144
5.29 Tang::GarbageCollected Class Reference . . . 144
5.29.1 Detailed Description . . . 146
5.29.2 Constructor & Destructor Documentation . . . 146
5.29.2.1 GarbageCollected() [1/3] . . . 146
5.29.2.2 GarbageCollected() [2/3] . . . 147
5.29.2.3 ~GarbageCollected() . . . 147
5.29.2.4 GarbageCollected() [3/3] . . . 147
5.29.3 Member Function Documentation . . . 147

5.29.3.1 make() . . . 147
5.29.3.2 operator"!() . . . 148
5.29.3.3 operator"!=() . . . 148
5.29.3.4 operator%() . . . 149
5.29.3.5 operator∗() [1/2] . . . 150
5.29.3.6 operator∗() [2/2] . . . 150
5.29.3.7 operator+() . . . 150
5.29.3.8 operator-() [1/2] . . . 151
5.29.3.9 operator-() [2/2] . . . 151
5.29.3.10 operator->() . . . 152
5.29.3.11 operator/() . . . 152
5.29.3.12 operator<() . . . 153
5.29.3.13 operator<=() . . . 153
5.29.3.14 operator=() [1/2] . . . 154
5.29.3.15 operator=() [2/2] . . . 154
5.29.3.16 operator==() [1/8] . . . 155
5.29.3.17 operator==() [2/8] . . . 155
5.29.3.18 operator==() [3/8] . . . 156
5.29.3.19 operator==() [4/8] . . . 156
5.29.3.20 operator==() [5/8] . . . 156
5.29.3.21 operator==() [6/8] . . . 157
5.29.3.22 operator==() [7/8] . . . 157
5.29.3.23 operator==() [8/8] . . . 157
5.29.3.24 operator>() . . . 159
5.29.3.25 operator>=() . . . 159
5.29.4 Friends And Related Function Documentation . . . 160
5.29.4.1 operator<< . . . 160
5.30 Tang::location Class Reference . . . 160
5.30.1 Detailed Description . . . 162
5.31 Tang::position Class Reference . . . 162
5.31.1 Detailed Description . . . 163
5.32 Tang::Program Class Reference . . . 163
5.32.1 Detailed Description . . . 165
5.32.2 Member Enumeration Documentation . . . 165
5.32.2.1 CodeType . . . 165
5.32.3 Constructor & Destructor Documentation . . . 165
5.32.3.1 Program() . . . 165
5.32.4 Member Function Documentation . . . 165
5.32.4.1 addBytecode() . . . 165
5.32.4.2 addIdentifier() . . . 167
5.32.4.3 addString() . . . 167
5.32.4.4 dumpBytecode() . . . 167

5.32.4.5 execute() . . . 168
5.32.4.6 getAst() . . . 168
5.32.4.7 getBytecode() . . . 168
5.32.4.8 getCode() . . . 169
5.32.4.9 getIdentifiers() . . . 169
5.32.4.10 getResult() . . . 169
5.32.4.11 getStrings() . . . 169
5.32.4.12 pushEnvironment() . . . 169
5.32.4.13 setFunctionStackDeclaration() . . . 170
5.32.4.14 setJumpTarget() . . . 170
5.32.5 Member Data Documentation . . . 170
5.32.5.1 functionsDeclared . . . 170
5.33 Tang::SingletonObjectPool< T > Class Template Reference . . . 171
5.33.1 Detailed Description . . . 171
5.33.2 Member Function Documentation . . . 171
5.33.2.1 get() . . . 172
5.33.2.2 getInstance() . . . 172
5.33.2.3 recycle() . . . 172
5.34 Tang::TangBase Class Reference . . . 172
5.34.1 Detailed Description . . . 173
5.34.2 Constructor & Destructor Documentation . . . 173
5.34.2.1 TangBase() . . . 173
5.34.3 Member Function Documentation . . . 173
5.34.3.1 compileScript() . . . 173
5.35 Tang::TangScanner Class Reference . . . 174
5.35.1 Detailed Description . . . 175
5.35.2 Constructor & Destructor Documentation . . . 175
5.35.2.1 TangScanner() . . . 175
5.35.3 Member Function Documentation . . . 175
5.35.3.1 get_next_token() . . . 175

6 File Documentation **177**

6.1 build/generated/location.hh File Reference . . . 177
6.1.1 Detailed Description . . . 178
6.1.2 Function Documentation . . . 178
6.1.2.1 operator<<() [1/2] . . . 178
6.1.2.2 operator<<() [2/2] . . . 179
6.2 include/astNode.hpp File Reference . . . 179
6.2.1 Detailed Description . . . 180
6.3 include/astNodeAssign.hpp File Reference . . . 180
6.3.1 Detailed Description . . . 181
6.4 include/astNodeBinary.hpp File Reference . . . 181

6.4.1 Detailed Description . . . 182
6.5 include/astNodeBlock.hpp File Reference . . . 182
6.5.1 Detailed Description . . . 183
6.6 include/astNodeBoolean.hpp File Reference . . . 183
6.6.1 Detailed Description . . . 184
6.7 include/astNodeCast.hpp File Reference . . . 184
6.7.1 Detailed Description . . . 185
6.8 include/astNodeDoWhile.hpp File Reference . . . 185
6.8.1 Detailed Description . . . 186
6.9 include/astNodeFloat.hpp File Reference . . . 186
6.9.1 Detailed Description . . . 187
6.10 include/astNodeFor.hpp File Reference . . . 187
6.10.1 Detailed Description . . . 188
6.11 include/astNodeFunctionCall.hpp File Reference . . . 188
6.11.1 Detailed Description . . . 189
6.12 include/astNodeFunctionDeclaration.hpp File Reference . . . 189
6.12.1 Detailed Description . . . 190
6.13 include/astNodeIdentifier.hpp File Reference . . . 190
6.13.1 Detailed Description . . . 191
6.14 include/astNodeIfElse.hpp File Reference . . . 191
6.14.1 Detailed Description . . . 192
6.15 include/astNodeInteger.hpp File Reference . . . 192
6.15.1 Detailed Description . . . 193
6.16 include/astNodePrint.hpp File Reference . . . 193
6.16.1 Detailed Description . . . 194
6.17 include/astNodeReturn.hpp File Reference . . . 194
6.17.1 Detailed Description . . . 195
6.18 include/astNodeString.hpp File Reference . . . 195
6.18.1 Detailed Description . . . 196
6.19 include/astNodeTernary.hpp File Reference . . . 196
6.19.1 Detailed Description . . . 197
6.20 include/astNodeUnary.hpp File Reference . . . 197
6.20.1 Detailed Description . . . 198
6.21 include/astNodeWhile.hpp File Reference . . . 198
6.21.1 Detailed Description . . . 199
6.22 include/computedExpression.hpp File Reference . . . 199
6.22.1 Detailed Description . . . 199
6.23 include/computedExpressionBoolean.hpp File Reference . . . 200
6.23.1 Detailed Description . . . 200
6.24 include/computedExpressionCompiledFunction.hpp File Reference . . . 201
6.24.1 Detailed Description . . . 201
6.25 include/computedExpressionError.hpp File Reference . . . 202

6.25.1 Detailed Description . . . 202
6.26 include/computedExpressionFloat.hpp File Reference . . . 203
6.26.1 Detailed Description . . . 203
6.27 include/computedExpressionInteger.hpp File Reference . . . 204
6.27.1 Detailed Description . . . 204
6.28 include/computedExpressionString.hpp File Reference . . . 205
6.28.1 Detailed Description . . . 205
6.29 include/error.hpp File Reference . . . 206
6.29.1 Detailed Description . . . 206
6.30 include/garbageCollected.hpp File Reference . . . 207
6.30.1 Detailed Description . . . 207
6.31 include/macros.hpp File Reference . . . 207
6.31.1 Detailed Description . . . 208
6.32 include/opcode.hpp File Reference . . . 208
6.32.1 Detailed Description . . . 208
6.32.2 Enumeration Type Documentation . . . 208
6.32.2.1 Opcode . . . 208
6.33 include/program.hpp File Reference . . . 209
6.33.1 Detailed Description . . . 210
6.34 include/singletonObjectPool.hpp File Reference . . . 210
6.34.1 Detailed Description . . . 211
6.35 include/tang.hpp File Reference . . . 211
6.35.1 Detailed Description . . . 212
6.36 include/tangBase.hpp File Reference . . . 212
6.36.1 Detailed Description . . . 213
6.37 include/tangScanner.hpp File Reference . . . 214
6.37.1 Detailed Description . . . 215
6.38 src/astNode.cpp File Reference . . . 215
6.38.1 Detailed Description . . . 215
6.39 src/astNodeAssign.cpp File Reference . . . 215
6.39.1 Detailed Description . . . 216
6.40 src/astNodeBinary.cpp File Reference . . . 216
6.40.1 Detailed Description . . . 217
6.41 src/astNodeBlock.cpp File Reference . . . 217
6.41.1 Detailed Description . . . 217
6.42 src/astNodeBoolean.cpp File Reference . . . 217
6.42.1 Detailed Description . . . 218
6.43 src/astNodeCast.cpp File Reference . . . 218
6.43.1 Detailed Description . . . 219
6.44 src/astNodeDoWhile.cpp File Reference . . . 219
6.44.1 Detailed Description . . . 219
6.45 src/astNodeFloat.cpp File Reference . . . 219

6.45.1 Detailed Description . . . 220
6.46 src/astNodeFor.cpp File Reference . . . 220
6.46.1 Detailed Description . . . 221
6.47 src/astNodeFunctionCall.cpp File Reference . . . 221
6.47.1 Detailed Description . . . 222
6.48 src/astNodeFunctionDeclaration.cpp File Reference . . . 222
6.48.1 Detailed Description . . . 223
6.49 src/astNodeIdentifier.cpp File Reference . . . 223
6.49.1 Detailed Description . . . 224
6.50 src/astNodeIfElse.cpp File Reference . . . 224
6.50.1 Detailed Description . . . 224
6.51 src/astNodeInteger.cpp File Reference . . . 224
6.51.1 Detailed Description . . . 225
6.52 src/astNodePrint.cpp File Reference . . . 225
6.52.1 Detailed Description . . . 226
6.53 src/astNodeReturn.cpp File Reference . . . 226
6.53.1 Detailed Description . . . 226
6.54 src/astNodeString.cpp File Reference . . . 226
6.54.1 Detailed Description . . . 227
6.55 src/astNodeTernary.cpp File Reference . . . 227
6.55.1 Detailed Description . . . 228
6.56 src/astNodeUnary.cpp File Reference . . . 228
6.56.1 Detailed Description . . . 229
6.57 src/astNodeWhile.cpp File Reference . . . 229
6.57.1 Detailed Description . . . 230
6.58 src/computedExpression.cpp File Reference . . . 230
6.58.1 Detailed Description . . . 231
6.59 src/computedExpressionBoolean.cpp File Reference . . . 231
6.59.1 Detailed Description . . . 232
6.60 src/computedExpressionCompiledFunction.cpp File Reference . . . 232
6.60.1 Detailed Description . . . 232
6.61 src/computedExpressionError.cpp File Reference . . . 232
6.61.1 Detailed Description . . . 233
6.62 src/computedExpressionFloat.cpp File Reference . . . 233
6.62.1 Detailed Description . . . 234
6.63 src/computedExpressionInteger.cpp File Reference . . . 234
6.63.1 Detailed Description . . . 234
6.64 src/computedExpressionString.cpp File Reference . . . 235
6.64.1 Detailed Description . . . 235
6.65 src/error.cpp File Reference . . . 235
6.65.1 Detailed Description . . . 236
6.65.2 Function Documentation . . . 236

6.65.2.1 operator<<() 236
6.66 src/program-dumpBytecode.cpp File Reference 237
6.66.1 Detailed Description 237
6.66.2 Macro Definition Documentation 237
6.66.2.1 DUMPPROGRAMCHECK 238
6.67 src/program-execute.cpp File Reference 238
6.67.1 Detailed Description 239
6.67.2 Macro Definition Documentation 239
6.67.2.1 EXECUTEPROGRAMCHECK 239
6.67.2.2 STACKCHECK 239
6.68 src/program.cpp File Reference 239
6.68.1 Detailed Description 240
6.69 src/tangBase.cpp File Reference 240
6.69.1 Detailed Description 241
6.70 test/test.cpp File Reference 241
6.70.1 Detailed Description 242
6.71 test/testGarbageCollected.cpp File Reference 242
6.71.1 Detailed Description 243
6.72 test/testSingletonObjectPool.cpp File Reference 243
6.72.1 Detailed Description 243

Index **245**

Chapter 1

Tang: A Template Language

1.1 Quick Description

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

- YouTube playlist
- GitHub repository

1.2 Features

The following features are planned:

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

1.3 License


```
MIT License

Copyright (c) 2022 Corey Pennycuff

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

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

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Tang::AstNode . . . 11
- Tang::AstNodeAssign . . . 15
- Tang::AstNodeBinary . . . 18
- Tang::AstNodeBlock . . . 22
- Tang::AstNodeBoolean . . . 25
- Tang::AstNodeCast . . . 27
- Tang::AstNodeDoWhile . . . 31
- Tang::AstNodeFloat . . . 34
- Tang::AstNodeFor . . . 37
- Tang::AstNodeFunctionCall . . . 40
- Tang::AstNodeFunctionDeclaration . . . 43
- Tang::AstNodeIdentifier . . . 46
- Tang::AstNodeIfElse . . . 49
- Tang::AstNodeInteger . . . 53
- Tang::AstNodePrint . . . 56
- Tang::AstNodeReturn . . . 59
- Tang::AstNodeString . . . 62
- Tang::AstNodeTernary . . . 66
- Tang::AstNodeUnary . . . 69
- Tang::AstNodeWhile . . . 72

Tang::ComputedExpression . . . 75
- Tang::ComputedExpressionBoolean . . . 84
- Tang::ComputedExpressionCompiledFunction . . . 93
- Tang::ComputedExpressionError . . . 104
- Tang::ComputedExpressionFloat . . . 113
- Tang::ComputedExpressionInteger . . . 123
- Tang::ComputedExpressionString . . . 132

Tang::Error . . . 142
Tang::GarbageCollected . . . 144
Tang::location . . . 160
Tang::position . . . 162
Tang::Program . . . 163
Tang::SingletonObjectPool< T > . . . 171
Tang::TangBase . . . 172
TangTangFlexLexer
- Tang::TangScanner . . . 174

Chapter 3

Class Index

3.1 Class List

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

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

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

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

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

Tang::AstNodeBoolean
: An AstNode that represents a boolean literal . . . 25

Tang::AstNodeCast
: An AstNode that represents a typecast of an expression . . . 27

Tang::AstNodeDoWhile
: An AstNode that represents a do..while statement . . . 31

Tang::AstNodeFloat
: An AstNode that represents an float literal . . . 34

Tang::AstNodeFor
: An AstNode that represents an if() statement . . . 37

Tang::AstNodeFunctionCall
: An AstNode that represents a function call . . . 40

Tang::AstNodeFunctionDeclaration
: An AstNode that represents a function declaration . . . 43

Tang::AstNodeIdentifier
: An AstNode that represents an identifier . . . 46

Tang::AstNodeIfElse
: An AstNode that represents an if..else statement . . . 49

Tang::AstNodeInteger
: An AstNode that represents an integer literal . . . 53

Tang::AstNodePrint
: An AstNode that represents a print typeeration . . . 56

Tang::AstNodeReturn
: An AstNode that represents a `return` statement . . . 59

Tang::AstNodeString
: An AstNode that represents a string literal . . . 62

Tang::AstNodeTernary
: An AstNode that represents a ternary expression . . . 66

Tang::AstNodeUnary
An AstNode that represents a unary negation . . . 69
Tang::AstNodeWhile
An AstNode that represents a while statement . . . 72
Tang::ComputedExpression
Represents the result of a computation that has been executed . . . 75
Tang::ComputedExpressionBoolean
Represents an Boolean that is the result of a computation . . . 84
Tang::ComputedExpressionCompiledFunction
Represents a Compiled Function declared in the script . . . 93
Tang::ComputedExpressionError
Represents a Runtime Error . . . 104
Tang::ComputedExpressionFloat
Represents a Float that is the result of a computation . . . 113
Tang::ComputedExpressionInteger
Represents an Integer that is the result of a computation . . . 123
Tang::ComputedExpressionString
Represents a String that is the result of a computation . . . 132
Tang::Error
Used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error . . . 142
Tang::GarbageCollected
A container that acts as a resource-counting garbage collector for the specified type . . . 144
Tang::location
Two points in a source file . . . 160
Tang::position
A point in a source file . . . 162
Tang::Program
Represents a compiled script or template that may be executed . . . 163
Tang::SingletonObjectPool< T >
A thread-safe, singleton object pool of the designated type . . . 171
Tang::TangBase
The base class for the Tang programming language . . . 172
Tang::TangScanner
The Flex lexer class for the main Tang language . . . 174

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 . . . 177
include/astNode.hpp
Declare the Tang::AstNode base class . . . 179
include/astNodeAssign.hpp
Declare the Tang::AstNodeAssign class . . . 180
include/astNodeBinary.hpp
Declare the Tang::AstNodeBinary class . . . 181
include/astNodeBlock.hpp
Declare the Tang::AstNodeBlock class . . . 182
include/astNodeBoolean.hpp
Declare the Tang::AstNodeBoolean class . . . 183
include/astNodeCast.hpp
Declare the Tang::AstNodeCast class . . . 184
include/astNodeDoWhile.hpp
Declare the Tang::AstNodeDoWhile class . . . 185
include/astNodeFloat.hpp
Declare the Tang::AstNodeFloat class . . . 186
include/astNodeFor.hpp
Declare the Tang::AstNodeFor class . . . 187
include/astNodeFunctionCall.hpp
Declare the Tang::AstNodeFunctionCall class . . . 188
include/astNodeFunctionDeclaration.hpp
Declare the Tang::AstNodeFunctionDeclaration class . . . 189
include/astNodeIdentifier.hpp
Declare the Tang::AstNodeIdentifier class . . . 190
include/astNodeIfElse.hpp
Declare the Tang::AstNodeIfElse class . . . 191
include/astNodeInteger.hpp
Declare the Tang::AstNodeInteger class . . . 192
include/astNodePrint.hpp
Declare the Tang::AstNodePrint class . . . 193
include/astNodeReturn.hpp
Declare the Tang::AstNodeReturn class . . . 194
include/astNodeString.hpp
Declare the Tang::AstNodeString class . . . 195

include/astNodeTernary.hpp
Declare the Tang::AstNodeTernary class . . . 196
include/astNodeUnary.hpp
Declare the Tang::AstNodeUnary class . . . 197
include/astNodeWhile.hpp
Declare the Tang::AstNodeWhile class . . . 198
include/computedExpression.hpp
Declare the Tang::ComputedExpression base class . . . 199
include/computedExpressionBoolean.hpp
Declare the Tang::ComputedExpressionBoolean class . . . 200
include/computedExpressionCompiledFunction.hpp
Declare the Tang::ComputedExpressionCompiledFunction class . . . 201
include/computedExpressionError.hpp
Declare the Tang::ComputedExpressionError class . . . 202
include/computedExpressionFloat.hpp
Declare the Tang::ComputedExpressionFloat class . . . 203
include/computedExpressionInteger.hpp
Declare the Tang::ComputedExpressionInteger class . . . 204
include/computedExpressionString.hpp
Declare the Tang::ComputedExpressionString class . . . 205
include/error.hpp
Declare the Tang::Error class used to describe syntax and runtime errors . . . 206
include/garbageCollected.hpp
Declare the Tang::GarbageCollected class . . . 207
include/macros.hpp
Contains generic macros . . . 207
include/opcode.hpp
Declare the Opcodes used in the Bytecode representation of a program . . . 208
include/program.hpp
Declare the Tang::Program class used to compile and execute source code . . . 209
include/singletonObjectPool.hpp
Declare the Tang::SingletonObjectPool class . . . 210
include/tang.hpp
Header file supplied for use by 3rd party code so that they can easily include all necessary headers . . . 211
include/tangBase.hpp
Declare the Tang::TangBase class used to interact with Tang . . . 212
include/tangScanner.hpp
Declare the Tang::TangScanner used to tokenize a Tang script . . . 214
src/astNode.cpp
Define the Tang::AstNode class . . . 215
src/astNodeAssign.cpp
Define the Tang::AstNodeAssign class . . . 215
src/astNodeBinary.cpp
Define the Tang::AstNodeBinary class . . . 216
src/astNodeBlock.cpp
Define the Tang::AstNodeBlock class . . . 217
src/astNodeBoolean.cpp
Define the Tang::AstNodeBoolean class . . . 217
src/astNodeCast.cpp
Define the Tang::AstNodeCast class . . . 218
src/astNodeDoWhile.cpp
Define the Tang::AstNodeDoWhile class . . . 219
src/astNodeFloat.cpp
Define the Tang::AstNodeFloat class . . . 219
src/astNodeFor.cpp
Define the Tang::AstNodeFor class . . . 220

src/astNodeFunctionCall.cpp
Define the Tang::AstNodeFunctionCall class . . . 221
src/astNodeFunctionDeclaration.cpp
Define the Tang::AstNodeFunctionDeclaration class . . . 222
src/astNodeIdentifier.cpp
Define the Tang::AstNodeIdentifier class . . . 223
src/astNodeIfElse.cpp
Define the Tang::AstNodeIfElse class . . . 224
src/astNodeInteger.cpp
Define the Tang::AstNodeInteger class . . . 224
src/astNodePrint.cpp
Define the Tang::AstNodePrint class . . . 225
src/astNodeReturn.cpp
Define the Tang::AstNodeReturn class . . . 226
src/astNodeString.cpp
Define the Tang::AstNodeString class . . . 226
src/astNodeTernary.cpp
Define the Tang::AstNodeTernary class . . . 227
src/astNodeUnary.cpp
Define the Tang::AstNodeUnary class . . . 228
src/astNodeWhile.cpp
Define the Tang::AstNodeWhile class . . . 229
src/computedExpression.cpp
Define the Tang::ComputedExpression class . . . 230
src/computedExpressionBoolean.cpp
Define the Tang::ComputedExpressionBoolean class . . . 231
src/computedExpressionCompiledFunction.cpp
Define the Tang::ComputedExpressionCompiledFunction class . . . 232
src/computedExpressionError.cpp
Define the Tang::ComputedExpressionError class . . . 232
src/computedExpressionFloat.cpp
Define the Tang::ComputedExpressionFloat class . . . 233
src/computedExpressionInteger.cpp
Define the Tang::ComputedExpressionInteger class . . . 234
src/computedExpressionString.cpp
Define the Tang::ComputedExpressionString class . . . 235
src/error.cpp
Define the Tang::Error class . . . 235
src/program-dumpBytecode.cpp
Define the Tang::Program::dumpBytecode method . . . 237
src/program-execute.cpp
Define the Tang::Program::execute method . . . 238
src/program.cpp
Define the Tang::Program class . . . 239
src/tangBase.cpp
Define the Tang::TangBase class . . . 240
test/test.cpp
Test the general language behaviors . . . 241
test/testGarbageCollected.cpp
Test the generic behavior of the Tang::GarbageCollected class . . . 242
test/testSingletonObjectPool.cpp
Test the generic behavior of the Tang::SingletonObjectPool class . . . 243

Chapter 5

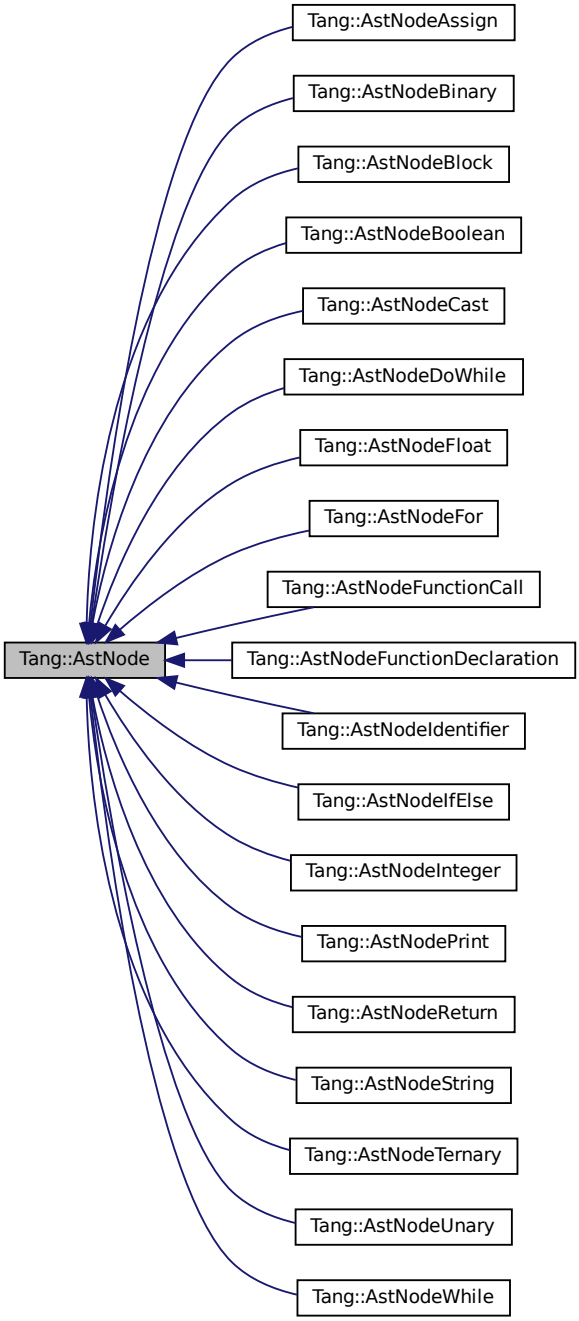
Class Documentation

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 compilePreprocess (Program &program) const

  Run any preprocess analysis needed before compilation.

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 compile()

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

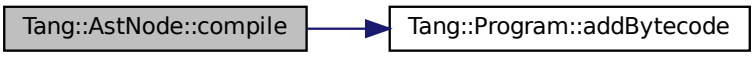
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeReturn, Tang::AstNodePrint, Tang::AstNodeInteger, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, 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:



5.1.3.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeReturn, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.1.3.3 dump()

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

Return a string that describes the contents of the node.

Parameters

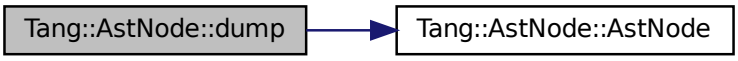
indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeReturn, Tang::AstNodePrint, Tang::AstNodeInteger, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, 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:

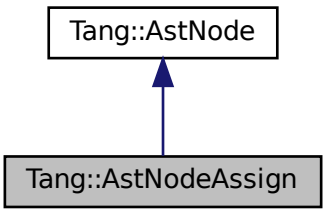
- include/astNode.hpp
- src/astNode.cpp

5.2 Tang::AstNodeAssign Class Reference

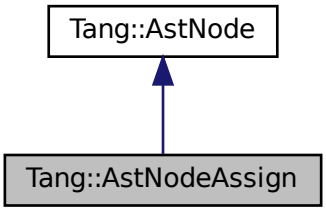
An AstNode that represents a binary expression.

`#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 compilePreprocess (Program &program) const override
  Run any preprocess analysis needed before compilation.

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 compile()

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

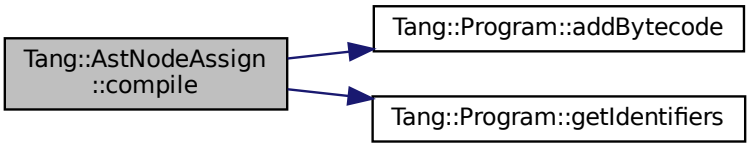
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.2.3.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.2.3.3 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:

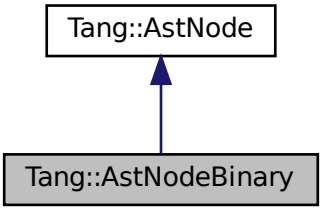
- include/astNodeAssign.hpp
- src/astNodeAssign.cpp

5.3 Tang::AstNodeBinary Class Reference

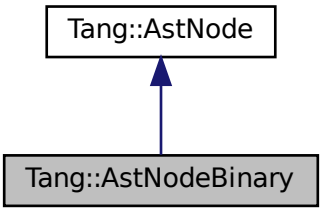
An AstNode that represents a binary expression.

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

  Run any preprocess analysis needed before compilation.

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 compile()

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

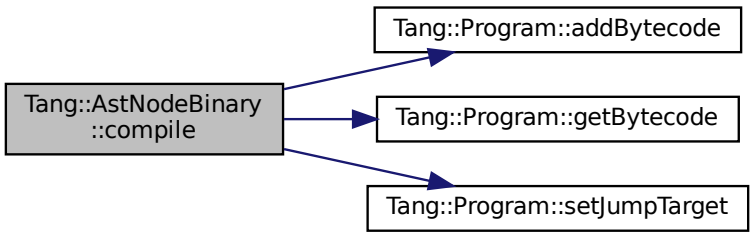
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.3.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.3.4.3 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:

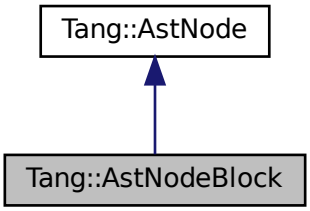
- include/astNodeBinary.hpp
- src/astNodeBinary.cpp

5.4 Tang::AstNodeBlock Class Reference

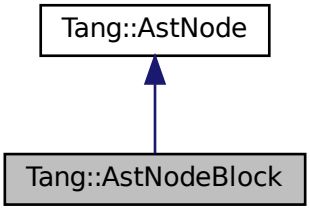
An AstNode that represents a code block.

`#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 compilePreprocess (Program &program) const override

  Run any preprocess analysis needed before compilation.

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 compile()

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

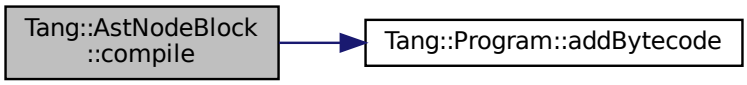
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.4.3.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.4.3.3 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:

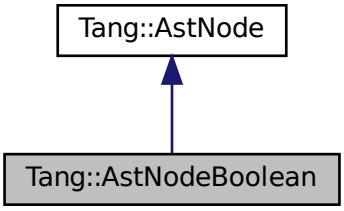
- include/astNodeBlock.hpp
- src/astNodeBlock.cpp

5.5 Tang::AstNodeBoolean Class Reference

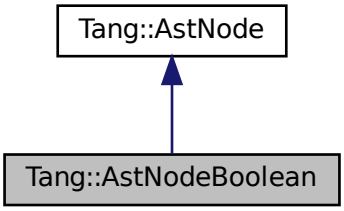
An AstNode that represents a boolean literal.

`#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 compilePreprocess (Program &program) const

  Run any preprocess analysis needed before compilation.

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 compile()

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

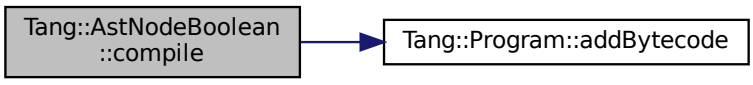
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.5.3.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeReturn, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.5.3.3 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:

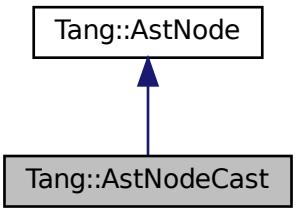
- include/astNodeBoolean.hpp
- src/astNodeBoolean.cpp

5.6 Tang::AstNodeCast Class Reference

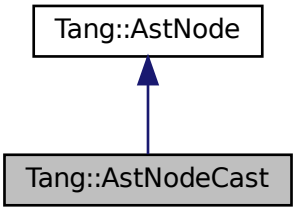
An AstNode that represents a typecast of an expression.

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

  Run any preprocess analysis needed before compilation.

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 compile()

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

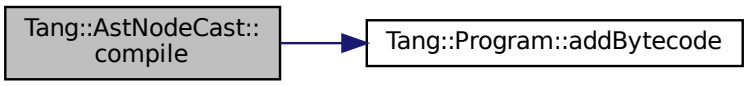
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.6.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.6.4.3 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:

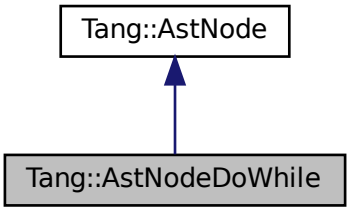
- include/astNodeCast.hpp
- src/astNodeCast.cpp

5.7 Tang::AstNodeDoWhile Class Reference

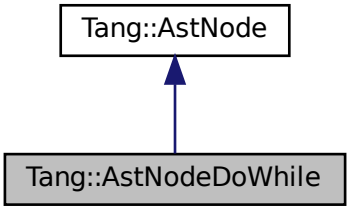
An AstNode that represents a do..while statement.

`#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 compilePreprocess (Program &program) const override

  Run any preprocess analysis needed before compilation.

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 compile()

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

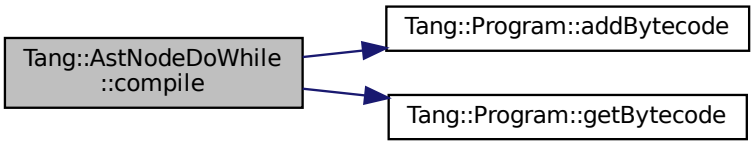
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.7.3.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.7.3.3 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:

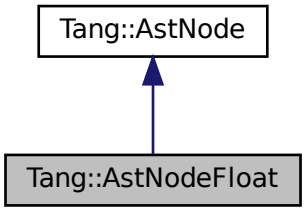
- include/astNodeDoWhile.hpp
- src/astNodeDoWhile.cpp

5.8 Tang::AstNodeFloat Class Reference

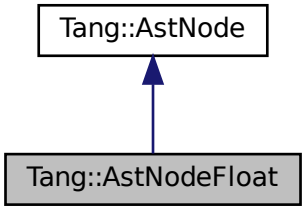
An AstNode that represents an float literal.

`#include <astNodeFloat.hpp>`

Inheritance diagram for Tang::AstNodeFloat:



Collaboration diagram for Tang::AstNodeFloat:



Public Member Functions

- AstNodeFloat (Tang::float_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 compilePreprocess (Program &program) const

  Run any preprocess analysis needed before compilation.

5.8.1 Detailed Description

An AstNode that represents an float literal.

Integers are represented by the `Tang::float_t` 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 (
            Tang::float_t 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 compile()

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

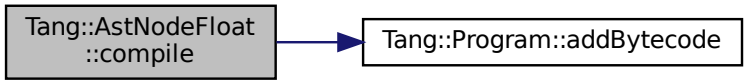
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.8.3.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeReturn, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.8.3.3 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:

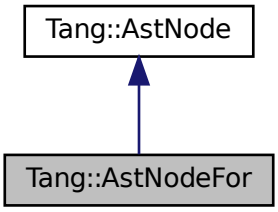
- include/astNodeFloat.hpp
- src/astNodeFloat.cpp

5.9 Tang::AstNodeFor Class Reference

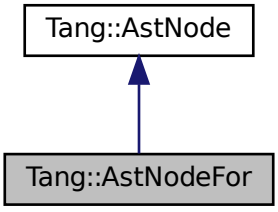
An AstNode that represents an if() statement.

`#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 compilePreprocess (Program &program) const override

  Run any preprocess analysis needed before compilation.

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 compile()

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

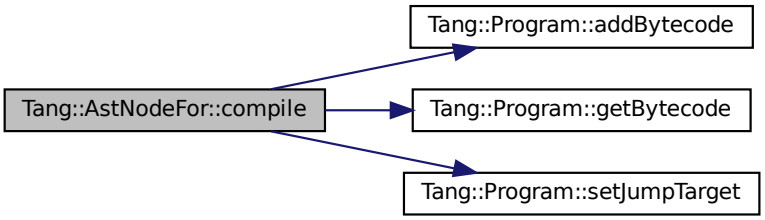
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.9.3.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.9.3.3 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:

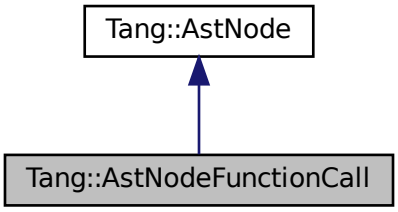
- include/astNodeFor.hpp
- src/astNodeFor.cpp

5.10 Tang::AstNodeFunctionCall Class Reference

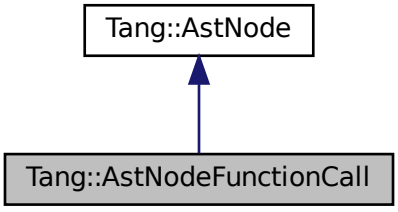
An AstNode that represents a function call.

`#include <astNodeFunctionCall.hpp>`

Inheritance diagram for Tang::AstNodeFunctionCall:



Collaboration diagram for Tang::AstNodeFunctionCall:



Public Member Functions

- AstNodeFunctionCall (std::shared_ptr< AstNode > function, std::vector< std::shared_ptr< AstNode >> argv, 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 compilePreprocess (Program &program) const override

  Run any preprocess analysis needed before compilation.

5.10.1 Detailed Description

An AstNode that represents a function call.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 AstNodeFunctionCall()

```
AstNodeFunctionCall::AstNodeFunctionCall (
            std::shared_ptr< AstNode > function,
            std::vector< std::shared_ptr< AstNode >> argv,
            Tang::location location )
```

The constructor.

Parameters

function	The function being invoked.
argv	The list of arguments provided to the function.
location	The location associated with the expression.

5.10.3 Member Function Documentation

5.10.3.1 compile()

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

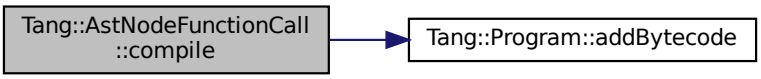
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.10.3.2 compilePreprocess()

```
void AstNodeFunctionCall::compilePreprocess (
            Program & program ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.10.3.3 dump()

```
string AstNodeFunctionCall::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:

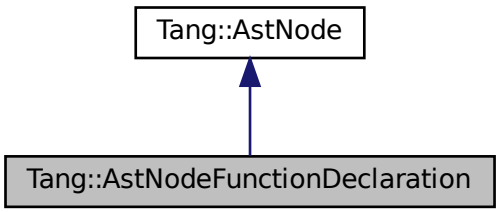
- include/astNodeFunctionCall.hpp
- src/astNodeFunctionCall.cpp

5.11 Tang::AstNodeFunctionDeclaration Class Reference

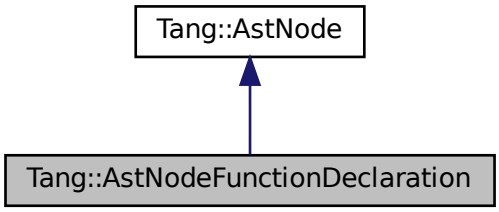
An AstNode that represents a function declaration.

`#include <astNodeFunctionDeclaration.hpp>`

Inheritance diagram for Tang::AstNodeFunctionDeclaration:



Collaboration diagram for Tang::AstNodeFunctionDeclaration:



Public Member Functions

- AstNodeFunctionDeclaration (std::string name, std::vector< std::string > arguments, 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 compilePreprocess (Program &program) const override

  Run any preprocess analysis needed before compilation.

5.11.1 Detailed Description

An AstNode that represents a function declaration.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 AstNodeFunctionDeclaration()

```
AstNodeFunctionDeclaration::AstNodeFunctionDeclaration (
            std::string name,
            std::vector< std::string > arguments,
            shared_ptr< AstNode > codeBlock,
            Tang::location location )
```

The constructor.

Parameters

name	The name of the function.
arguments	The arguments expected to be provided.
codeBlock	The code executed as part of the function.
location	The location associated with the function declaration.

5.11.3 Member Function Documentation

5.11.3.1 compile()

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

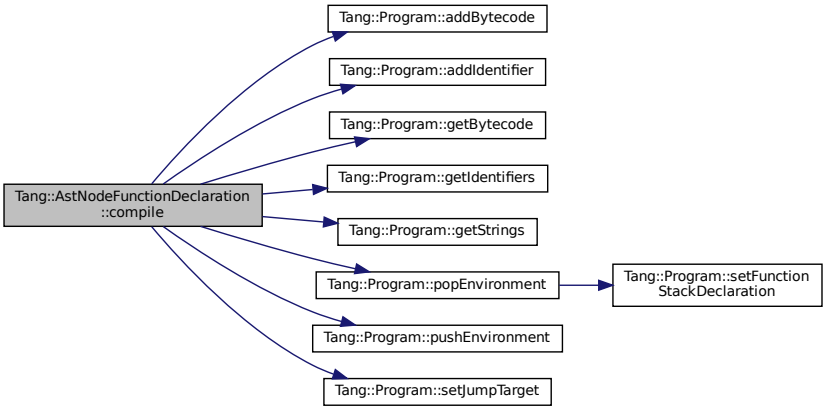
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.11.3.2 compilePreprocess()

```
void AstNodeFunctionDeclaration::compilePreprocess (
            Program & program ) const  [override], [virtual]
```

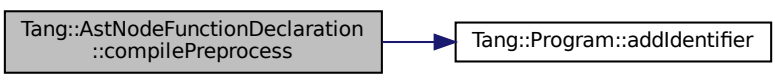
Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.11.3.3 dump()

```
string AstNodeFunctionDeclaration::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:

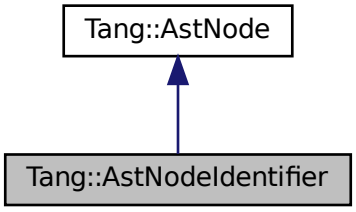
- include/astNodeFunctionDeclaration.hpp
- src/astNodeFunctionDeclaration.cpp

5.12 Tang::AstNodeIdentifier Class Reference

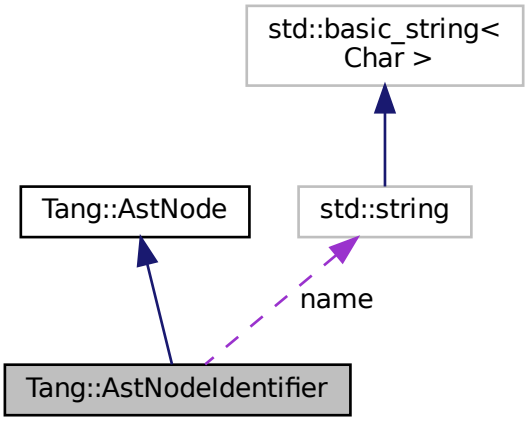
An AstNode that represents an identifier.

`#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 compilePreprocess (Program &program) const override
  Run any preprocess analysis needed before compilation.

Public Attributes

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

5.12.1 Detailed Description

An AstNode that represents an identifier.

Identifier names are represented by a string.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 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.

5.12.3 Member Function Documentation

5.12.3.1 compile()

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

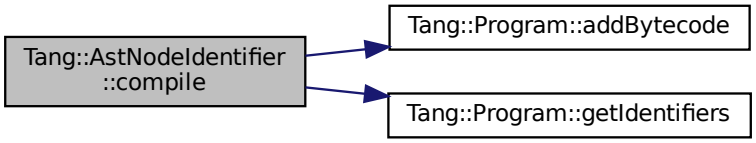
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.12.3.2 compilePreprocess()

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

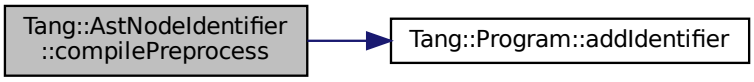
Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.12.3.3 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:

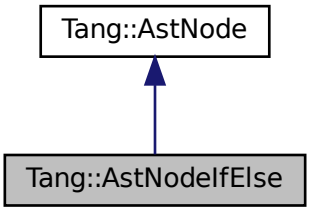
- include/astNodeIdentifier.hpp
- src/astNodeIdentifier.cpp

5.13 Tang::AstNodeIfElse Class Reference

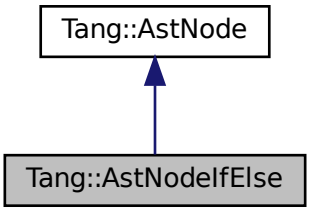
An AstNode that represents an if..else statement.

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

  Run any preprocess analysis needed before compilation.

5.13.1 Detailed Description

An AstNode that represents an if..else statement.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 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.

5.13.2.2 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.

5.13.3 Member Function Documentation

5.13.3.1 compile()

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

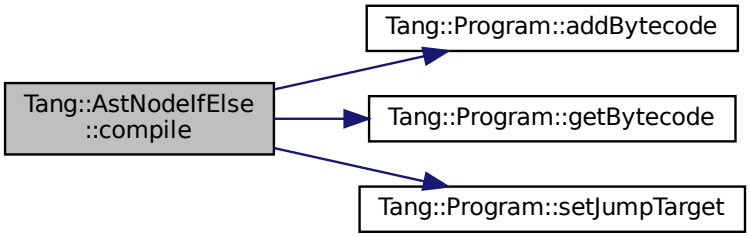
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.13.3.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.13.3.3 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:

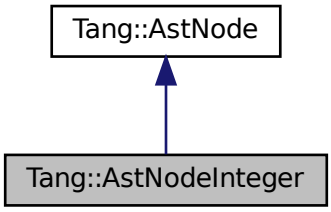
- include/astNodeIfElse.hpp
- src/astNodeIfElse.cpp

5.14 Tang::AstNodeInteger Class Reference

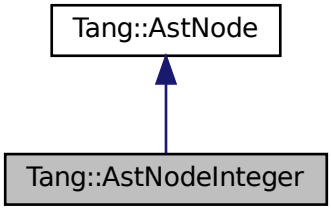
An AstNode that represents an integer literal.

`#include <astNodeInteger.hpp>`

Inheritance diagram for Tang::AstNodeInteger:



Collaboration diagram for Tang::AstNodeInteger:



Public Member Functions

- AstNodeInteger (Tang::integer_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 compilePreprocess (Program &program) const

  Run any preprocess analysis needed before compilation.

5.14.1 Detailed Description

An AstNode that represents an integer literal.

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

5.14.2 Constructor & Destructor Documentation

5.14.2.1 AstNodeInteger()

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

The constructor.

Parameters

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

5.14.3 Member Function Documentation

5.14.3.1 compile()

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

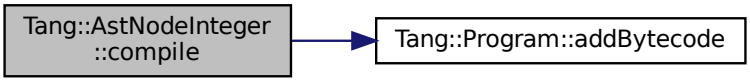
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.14.3.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeReturn, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.14.3.3 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:

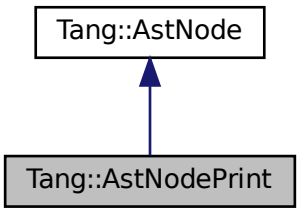
- include/astNodeInteger.hpp
- src/astNodeInteger.cpp

5.15 Tang::AstNodePrint Class Reference

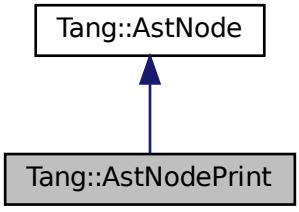
An AstNode that represents a print typeeration.

`#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 compilePreprocess (Program &program) const override

  Run any preprocess analysis needed before compilation.

5.15.1 Detailed Description

An AstNode that represents a print typeeration.

5.15.2 Member Enumeration Documentation

5.15.2.1 Type

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

The type of print() requested.

Enumerator

Default	Use the default print.

5.15.3 Constructor & Destructor Documentation

5.15.3.1 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.

5.15.4 Member Function Documentation

5.15.4.1 compile()

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

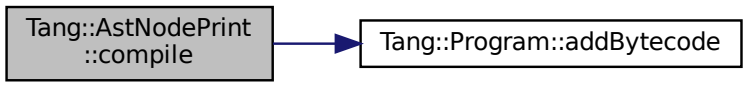
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.15.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.15.4.3 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:

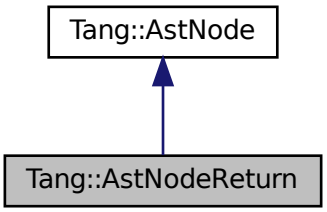
- include/astNodePrint.hpp
- src/astNodePrint.cpp

5.16 Tang::AstNodeReturn Class Reference

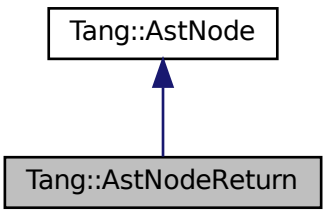
An AstNode that represents a `return` statement.

`#include <astNodeReturn.hpp>`

Inheritance diagram for Tang::AstNodeReturn:



Collaboration diagram for Tang::AstNodeReturn:



Public Member Functions

- AstNodeReturn (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 compilePreprocess (Program &program) const override

  Run any preprocess analysis needed before compilation.

5.16.1 Detailed Description

An AstNode that represents a `return` statement.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 AstNodeReturn()

```
AstNodeReturn::AstNodeReturn (
            shared_ptr< AstNode > expression,
            Tang::location location )
```

The constructor.

Parameters

expression	The expression to be returned.
location	The location associated with the return statement.

5.16.3 Member Function Documentation

5.16.3.1 compile()

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

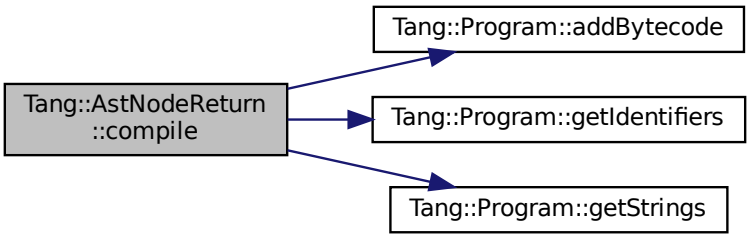
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.16.3.2 compilePreprocess()

```
void AstNodeReturn::compilePreprocess (
            Program & program ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.16.3.3 dump()

```
string AstNodeReturn::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:

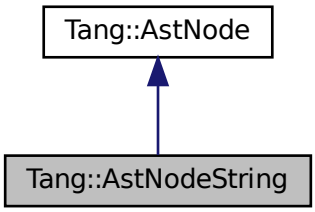
- include/astNodeReturn.hpp
- src/astNodeReturn.cpp

5.17 Tang::AstNodeString Class Reference

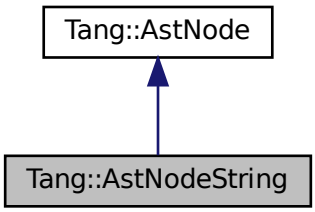
An AstNode that represents a string literal.

`#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 compilePreprocess (Program &program) const override

  Run any preprocess analysis needed before compilation.
- void compileLiteral (Tang::Program &program) const

  Compile the string and push it onto the stack.

5.17.1 Detailed Description

An AstNode that represents a string literal.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 AstNodeString()

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

The constructor.

Parameters

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

5.17.3 Member Function Documentation

5.17.3.1 compile()

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

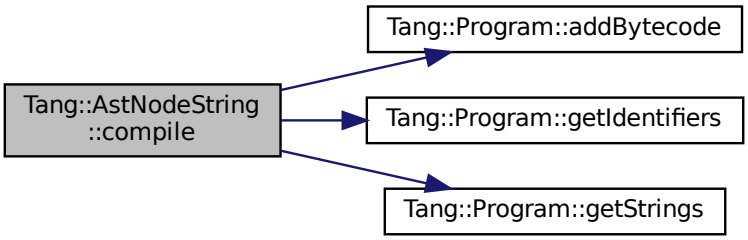
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.17.3.2 compileLiteral()

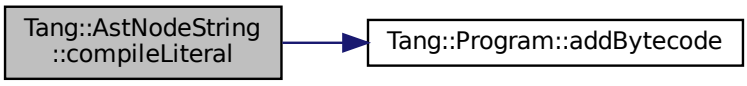
```
void AstNodeString::compileLiteral (
            Tang::Program & program ) const
```

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:



5.17.3.3 compilePreprocess()

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

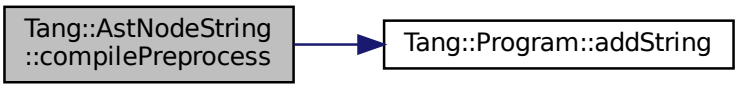
Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.17.3.4 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:

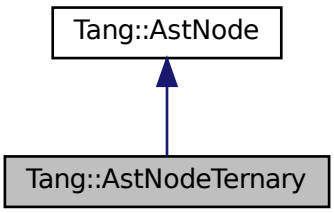
- include/astNodeString.hpp
- src/astNodeString.cpp

5.18 Tang::AstNodeTernary Class Reference

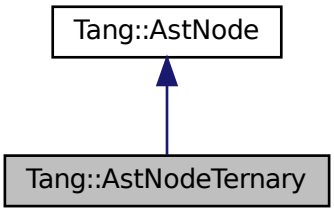
An AstNode that represents a ternary expression.

`#include <astNodeTernary.hpp>`

Inheritance diagram for Tang::AstNodeTernary:



Collaboration diagram for Tang::AstNodeTernary:



Public Member Functions

- AstNodeTernary (shared_ptr< AstNode > condition, shared_ptr< AstNode > trueExpression, shared_ptr< AstNode > falseExpression, 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 compilePreprocess (Program &program) const override

  Run any preprocess analysis needed before compilation.

5.18.1 Detailed Description

An AstNode that represents a ternary expression.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 AstNodeTernary()

```
AstNodeTernary::AstNodeTernary (
            shared_ptr< AstNode > condition,
            shared_ptr< AstNode > trueExpression,
            shared_ptr< AstNode > falseExpression,
            Tang::location location )
```

The constructor.

Parameters

condition	The expression which determines whether the trueExpression or falseExpression is executed.
trueExpression	The expression executed when the condition is true.
falseExpression	The expression executed when the condition is false.
location	The location associated with the expression.

5.18.3 Member Function Documentation

5.18.3.1 compile()

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

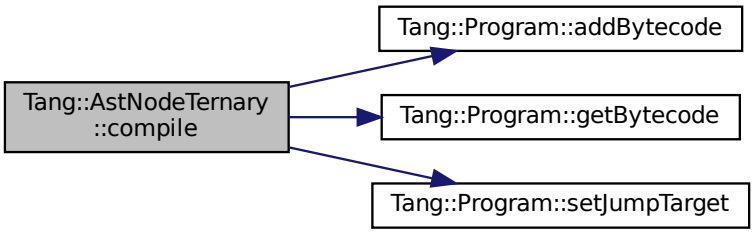
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.18.3.2 compilePreprocess()

```
void AstNodeTernary::compilePreprocess (
            Program & program ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.18.3.3 dump()

```
string AstNodeTernary::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:

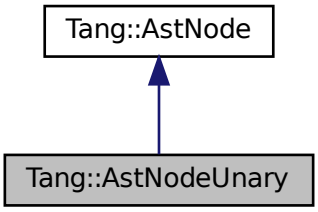
- include/astNodeTernary.hpp
- src/astNodeTernary.cpp

5.19 Tang::AstNodeUnary Class Reference

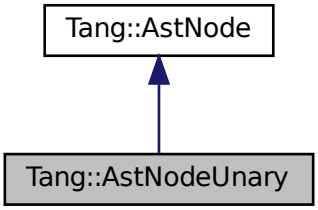
An AstNode that represents a unary negation.

`#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 compilePreprocess (Program &program) const override

  Run any preprocess analysis needed before compilation.

5.19.1 Detailed Description

An AstNode that represents a unary negation.

5.19.2 Member Enumeration Documentation

5.19.2.1 Operator

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

The type of operation.

Enumerator

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

5.19.3 Constructor & Destructor Documentation

5.19.3.1 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.

5.19.4 Member Function Documentation

5.19.4.1 compile()

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

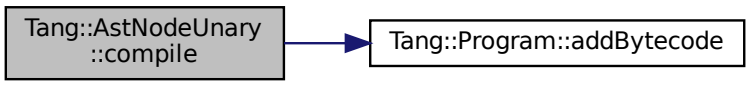
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.19.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.19.4.3 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:

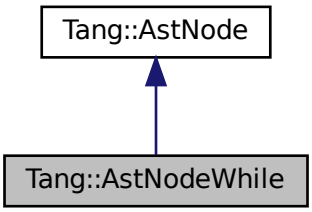
- include/astNodeUnary.hpp
- src/astNodeUnary.cpp

5.20 Tang::AstNodeWhile Class Reference

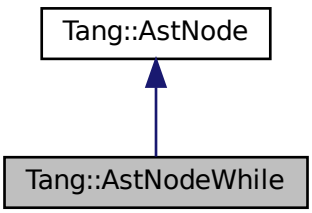
An AstNode that represents a while statement.

`#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 compilePreprocess (Program &program) const override

  Run any preprocess analysis needed before compilation.

5.20.1 Detailed Description

An AstNode that represents a while statement.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 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.

5.20.3 Member Function Documentation

5.20.3.1 compile()

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

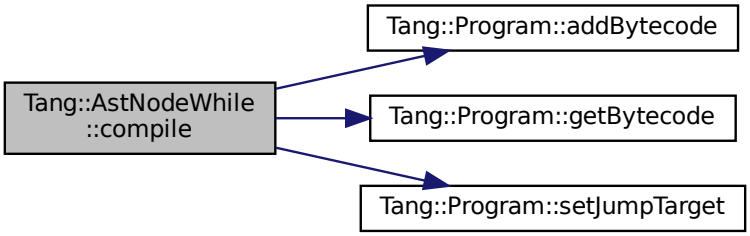
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.20.3.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.20.3.3 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:

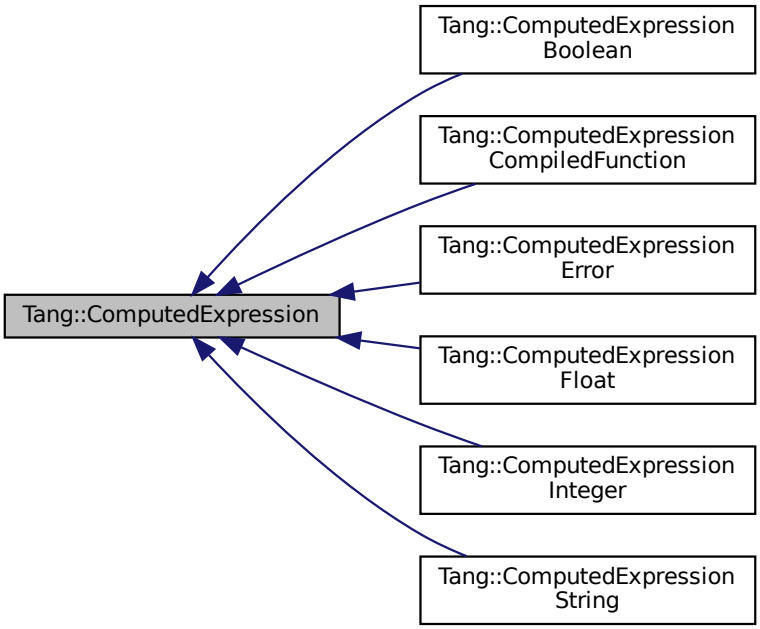
- include/astNodeWhile.hpp
- src/astNodeWhile.cpp

5.21 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 Tang::integer_t &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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.

5.21.1 Detailed Description

Represents the result of a computation that has been executed.

By default, it will represent a NULL value.

5.21.2 Member Function Documentation

5.21.2.1 __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.

5.21.2.2 __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.

5.21.2.3 __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.

5.21.2.4 __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, Tang::ComputedExpressionCompiledFunction, and Tang::ComputedExpressionBoolean.

5.21.2.5 __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.

5.21.2.6 __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.

5.21.2.7 __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.

5.21.2.8 __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.

5.21.2.9 __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.

5.21.2.10 __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.

5.21.2.11 __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.

5.21.2.12 __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.

5.21.2.13 __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.

5.21.2.14 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, Tang::ComputedExpressionCompiledFunction, and Tang::ComputedExpressionBoolean.

5.21.2.15 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.

5.21.2.16 is_equal() [2/6]

```
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.

5.21.2.17 is_equal() [3/6]

```
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.

5.21.2.18 is_equal() [4/6]

```
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.

5.21.2.19 is_equal() [5/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::float_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.

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

5.21.2.20 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::integer_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.

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

5.21.2.21 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, Tang::ComputedExpressionCompiledFunction, and Tang::ComputedExpressionBoolean.

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

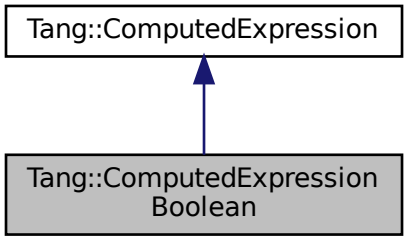
- include/computedExpression.hpp
- src/computedExpression.cpp

5.22 Tang::ComputedExpressionBoolean Class Reference

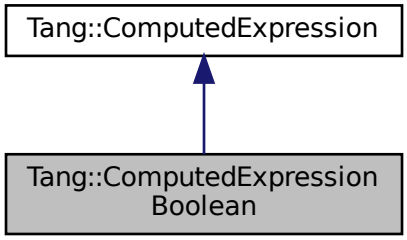
Represents an Boolean that is the result of a computation.

`#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 Tang::integer_t &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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.

5.22.1 Detailed Description

Represents an Boolean that is the result of a computation.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 ComputedExpressionBoolean()

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

Construct an Boolean result.

Parameters

val	The boolean value.

5.22.3 Member Function Documentation

5.22.3.1 __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.

5.22.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.3 __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 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.

5.22.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.7 __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.

5.22.3.8 __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 ComputedExpressionBoolean::__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 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.

5.22.3.13 __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 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.

5.22.3.15 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.

5.22.3.16 is_equal() [2/6]

```
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.

5.22.3.17 is_equal() [3/6]

```
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.

5.22.3.18 is_equal() [4/6]

```
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.

5.22.3.19 is_equal() [5/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::float_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.

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

5.22.3.20 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::integer_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.

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

5.22.3.21 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:

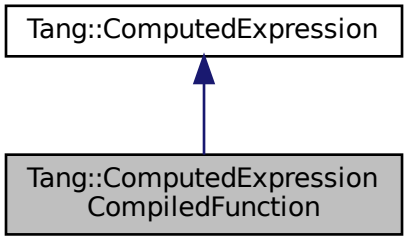
- include/computedExpressionBoolean.hpp
- src/computedExpressionBoolean.cpp

5.23 Tang::ComputedExpressionCompiledFunction Class Reference

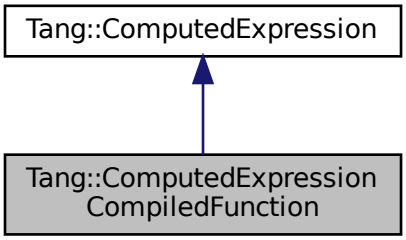
Represents a Compiled Function declared in the script.

```
#include <computedExpressionCompiledFunction.hpp>
```

Inheritance diagram for Tang::ComputedExpressionCompiledFunction:



Collaboration diagram for Tang::ComputedExpressionCompiledFunction:



Public Member Functions

- ComputedExpressionCompiledFunction (uint32_t argc, Tang::integer_t pc)

  Construct an CompiledFunction.
- 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 GarbageCollected __equal (const GarbageCollected &rhs) const override

  Perform an equalit test.
- uint32_t getArgc () const

  Get the `argc` value.
- Tang::integer_t getPc () const

  Get the bytecode target.
- virtual bool is_equal (const Tang::integer_t &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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 __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.

5.23.1 Detailed Description

Represents a Compiled Function declared in the script.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 ComputedExpressionCompiledFunction()

```
ComputedExpressionCompiledFunction::ComputedExpressionCompiledFunction (
            uint32_t argc,
            Tang::integer_t pc )
```

Construct an CompiledFunction.

Parameters

argc	The count of arguments that this function expects.
pc	The bytecode address of the start of the function.

5.23.3 Member Function Documentation

5.23.3.1 __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.

5.23.3.2 __boolean()

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

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.

5.23.3.3 __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.

5.23.3.4 __equal()

```
GarbageCollected ComputedExpressionCompiledFunction::__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.

5.23.3.5 __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.

5.23.3.6 __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.

5.23.3.7 __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.

5.23.3.8 __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.

5.23.3.9 __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.

5.23.3.10 __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.

5.23.3.11 __not()

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

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.

5.23.3.12 __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.

5.23.3.13 __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.

5.23.3.14 dump()

```
string ComputedExpressionCompiledFunction::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.

5.23.3.15 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.

5.23.3.16 is_equal() [2/6]

```
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.

5.23.3.17 is_equal() [3/6]

```
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.

5.23.3.18 is_equal() [4/6]

```
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.

5.23.3.19 is_equal() [5/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::float_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.

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

5.23.3.20 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::integer_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.

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

5.23.3.21 makeCopy()

```
GarbageCollected ComputedExpressionCompiledFunction::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:

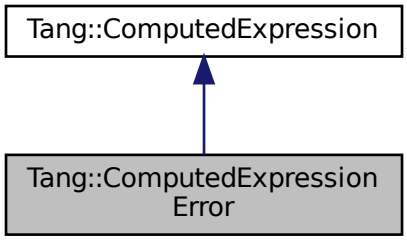
- include/computedExpressionCompiledFunction.hpp
- src/computedExpressionCompiledFunction.cpp

5.24 Tang::ComputedExpressionError Class Reference

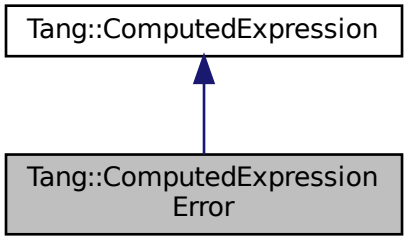
Represents a Runtime Error.

`#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 Tang::integer_t &val) const
  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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.

5.24.1 Detailed Description

Represents a Runtime Error.

5.24.2 Constructor & Destructor Documentation

5.24.2.1 ComputedExpressionError()

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

Construct a Runtime Error.

Parameters

error	The Tang::Error object.

5.24.3 Member Function Documentation

5.24.3.1 __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.

5.24.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.24.3.3 __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.

5.24.3.4 __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.

5.24.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.24.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.24.3.7 __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.

5.24.3.8 __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.

5.24.3.9 __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.

5.24.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.24.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.24.3.12 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.24.3.13 __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.

5.24.3.14 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.

5.24.3.15 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.

5.24.3.16 is_equal() [2/6]

```
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.

5.24.3.17 is_equal() [3/6]

```
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.

5.24.3.18 is_equal() [4/6]

```
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.

5.24.3.19 is_equal() [5/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::float_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.

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

5.24.3.20 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::integer_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.

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

5.24.3.21 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:

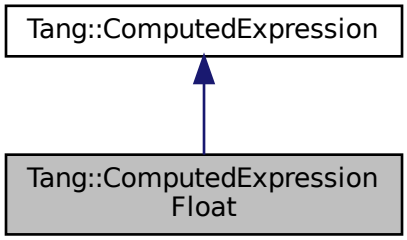
- include/computedExpressionError.hpp
- src/computedExpressionError.cpp

5.25 Tang::ComputedExpressionFloat Class Reference

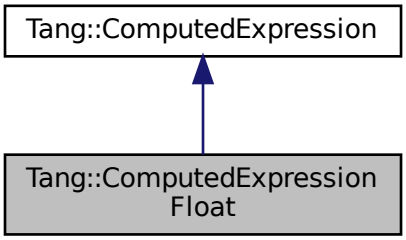
Represents a Float that is the result of a computation.

`#include <computedExpressionFloat.hpp>`

Inheritance diagram for Tang::ComputedExpressionFloat:



Collaboration diagram for Tang::ComputedExpressionFloat:



Public Member Functions

- ComputedExpressionFloat (Tang::float_t 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 Tang::integer_t &val) const override

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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**

5.25.1 Detailed Description

Represents a Float that is the result of a computation.

5.25.2 Constructor & Destructor Documentation

5.25.2.1 ComputedExpressionFloat()

```
ComputedExpressionFloat::ComputedExpressionFloat (
            Tang::float_t val )
```

Construct a Float result.

Parameters

val	The float value.

5.25.3 Member Function Documentation

5.25.3.1 __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.

5.25.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.25.3.3 __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.

5.25.3.4 __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.

5.25.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.25.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.25.3.7 __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.

5.25.3.8 __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.

5.25.3.9 __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.

5.25.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.25.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.25.3.12 __string()

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

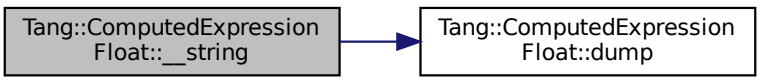
Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.25.3.13 __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.

5.25.3.14 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.

5.25.3.15 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.

5.25.3.16 is_equal() [2/6]

```
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.

5.25.3.17 is_equal() [3/6]

```
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.

5.25.3.18 is_equal() [4/6]

```
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.

5.25.3.19 is_equal() [5/6]

```
bool ComputedExpressionFloat::is_equal (
            const Tang::float_t & 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.

5.25.3.20 is_equal() [6/6]

```
bool ComputedExpressionFloat::is_equal (
            const Tang::integer_t & 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.

5.25.3.21 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:

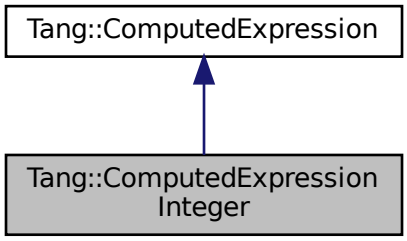
- include/computedExpressionFloat.hpp
- src/computedExpressionFloat.cpp

5.26 Tang::ComputedExpressionInteger Class Reference

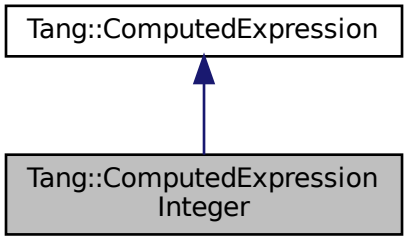
Represents an Integer that is the result of a computation.

`#include <computedExpressionInteger.hpp>`

Inheritance diagram for Tang::ComputedExpressionInteger:



Collaboration diagram for Tang::ComputedExpressionInteger:



Public Member Functions

- ComputedExpressionInteger (Tang::integer_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 Tang::integer_t &val) const override

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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**

5.26.1 Detailed Description

Represents an Integer that is the result of a computation.

5.26.2 Constructor & Destructor Documentation

5.26.2.1 ComputedExpressionInteger()

```
ComputedExpressionInteger::ComputedExpressionInteger (
            Tang::integer_t val )
```

Construct an Integer result.

Parameters

val	The integer value.

5.26.3 Member Function Documentation

5.26.3.1 __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.

5.26.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.26.3.3 __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.

5.26.3.4 __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.

5.26.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.26.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.26.3.7 __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.

5.26.3.8 __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.

5.26.3.9 __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.

5.26.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.26.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.26.3.12 __string()

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

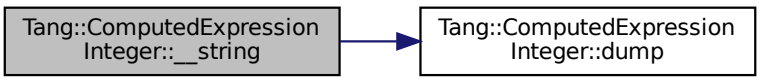
Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.26.3.13 __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.

5.26.3.14 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.

5.26.3.15 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.

5.26.3.16 is_equal() [2/6]

```
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.

5.26.3.17 is_equal() [3/6]

```
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.

5.26.3.18 is_equal() [4/6]

```
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.

5.26.3.19 is_equal() [5/6]

```
bool ComputedExpressionInteger::is_equal (
            const Tang::float_t & 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.

5.26.3.20 is_equal() [6/6]

```
bool ComputedExpressionInteger::is_equal (
            const Tang::integer_t & 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.

5.26.3.21 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:

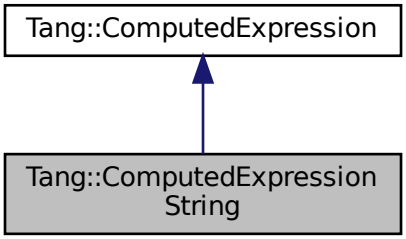
- include/computedExpressionInteger.hpp
- src/computedExpressionInteger.cpp

5.27 Tang::ComputedExpressionString Class Reference

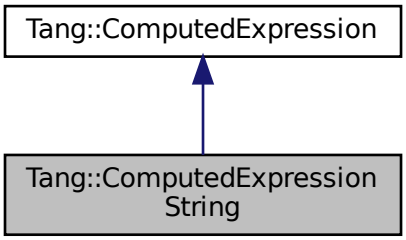
Represents a String that is the result of a computation.

```
#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 Tang::integer_t &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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.

5.27.1 Detailed Description

Represents a String that is the result of a computation.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 ComputedExpressionString()

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

Construct a String result.

Parameters

val	The string value.

5.27.3 Member Function Documentation

5.27.3.1 __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.

5.27.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.27.3.3 __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.

5.27.3.4 __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.

5.27.3.5 __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.

5.27.3.6 __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.

5.27.3.7 __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.

5.27.3.8 __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.

5.27.3.9 __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.

5.27.3.10 __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.

5.27.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.27.3.12 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.27.3.13 __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.

5.27.3.14 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.

5.27.3.15 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.

5.27.3.16 is_equal() [2/6]

```
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.

5.27.3.17 is_equal() [3/6]

```
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.

5.27.3.18 is_equal() [4/6]

```
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.

5.27.3.19 is_equal() [5/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::float_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.

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

5.27.3.20 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::integer_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.

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

5.27.3.21 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:

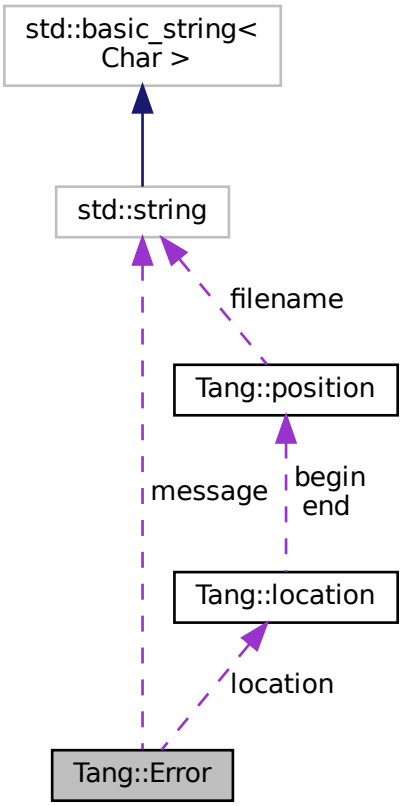
- include/computedExpressionString.hpp
- src/computedExpressionString.cpp

5.28 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.

5.28.1 Detailed Description

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

5.28.2 Constructor & Destructor Documentation

5.28.2.1 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.

5.28.2.2 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.

5.28.3 Friends And Related Function Documentation

5.28.3.1 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:

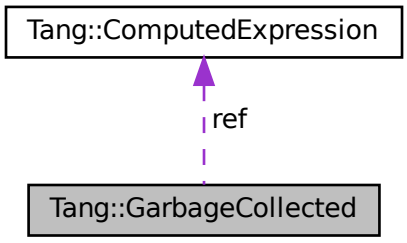
- include/error.hpp
- src/error.cpp

5.29 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 Tang::integer_t &val) const

  Compare the GarbageCollected tracked object with a supplied value.
- bool operator== (const Tang::float_t &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.

5.29.1 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.

5.29.2 Constructor & Destructor Documentation

5.29.2.1 GarbageCollected() [1/3]

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

Copy Constructor.

Parameters

The	other GarbageCollected object to copy.

5.29.2.2 GarbageCollected() [2/3]

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

Move Constructor.

Parameters

The	other GarbageCollected object to move.

5.29.2.3 ~GarbageCollected()

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

Destructor.

Clean up the tracked object, if appropriate.

5.29.2.4 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.

5.29.3 Member Function Documentation

5.29.3.1 make()

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

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

Creates a garbage-collected object of the specified type.

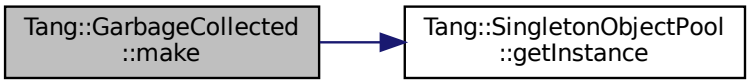
Parameters

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

Returns

A GarbageCollected object.

Here is the call graph for this function:



5.29.3.2 operator"!()

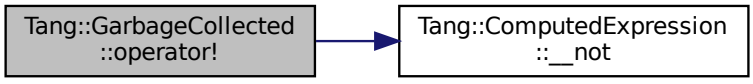
```
GarbageCollected GarbageCollected::operator!  ( ) const
```

Perform a logical not on the GarbageCollected value.

Returns

The result of the operation.

Here is the call graph for this function:



5.29.3.3 operator"!=()

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

Perform a != between two GarbageCollected values.

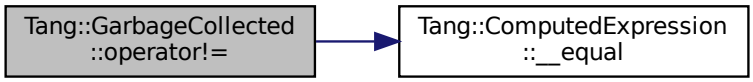
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.29.3.4 operator%()

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

Perform a modulo between two GarbageCollected values.

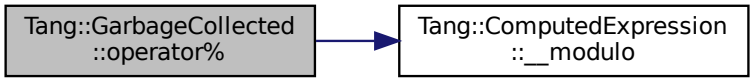
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.29.3.5 operator*() [1/2]

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

Access the tracked object.

Returns

A reference to the tracked object.

5.29.3.6 operator*() [2/2]

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

Perform a multiplication between two GarbageCollected values.

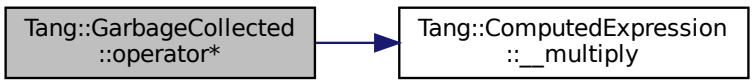
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.29.3.7 operator+()

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

Perform an addition between two GarbageCollected values.

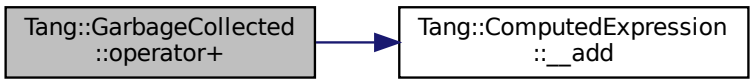
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.29.3.8 operator-() [1/2]

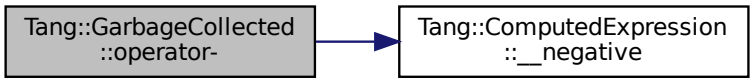
```
GarbageCollected GarbageCollected::operator- ( ) const
```

Perform a negation on the GarbageCollected value.

Returns

The result of the operation.

Here is the call graph for this function:



5.29.3.9 operator-() [2/2]

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

Perform a subtraction between two GarbageCollected values.

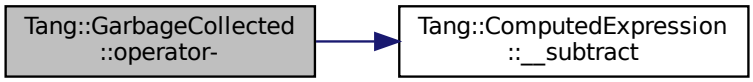
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.29.3.10 operator->()

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

Access the tracked object as a pointer.

Returns

A pointer to the tracked object.

5.29.3.11 operator/()

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

Perform a division between two GarbageCollected values.

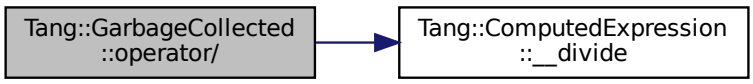
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.29.3.12 operator<()

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

Perform a < between two GarbageCollected values.

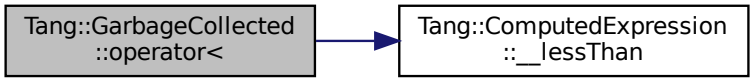
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.29.3.13 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.

5.29.3.14 operator=() [1/2]

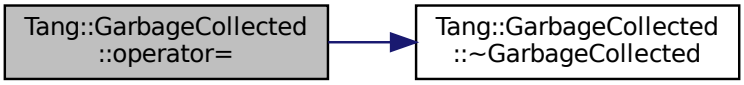
```
GarbageCollected& Tang::GarbageCollected::operator= (
            const GarbageCollected & other )  [inline]
```

Copy Assignment.

Parameters

The	other GarbageCollected object.

Here is the call graph for this function:



5.29.3.15 operator=() [2/2]

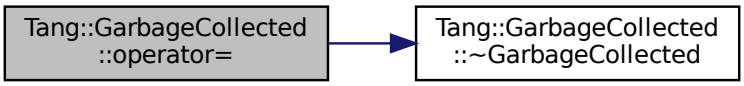
```
GarbageCollected& Tang::GarbageCollected::operator= (
            GarbageCollected && other )  [inline]
```

Move Assignment.

Parameters

The	other GarbageCollected object.

Here is the call graph for this function:



5.29.3.16 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.

5.29.3.17 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.

5.29.3.18 operator==() [3/8]

```
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.

5.29.3.19 operator==() [4/8]

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

Perform a == between two GarbageCollected values.

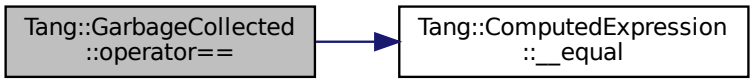
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.29.3.20 operator==() [5/8]

```
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.

5.29.3.21 operator==() [6/8]

```
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.

5.29.3.22 operator==() [7/8]

```
bool GarbageCollected::operator== (
            const Tang::float_t & 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.

5.29.3.23 operator==() [8/8]

```
bool GarbageCollected::operator== (
            const Tang::integer_t & 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.

5.29.3.24 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.

5.29.3.25 operator>=()

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

Perform a >= between two GarbageCollected values.

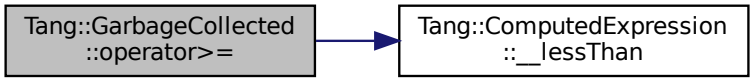
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.29.4 Friends And Related Function Documentation

5.29.4.1 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:

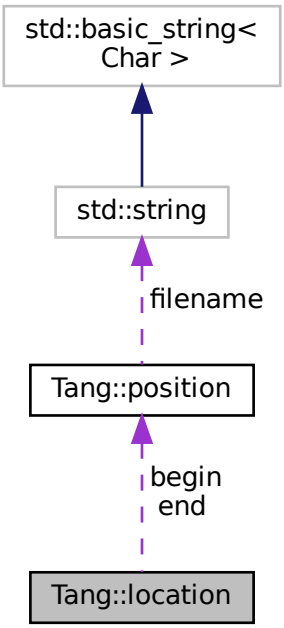
- include/garbageCollected.hpp
- src/garbageCollected.cpp

5.30 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.

5.30.1 Detailed Description

Two points in a source file.

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

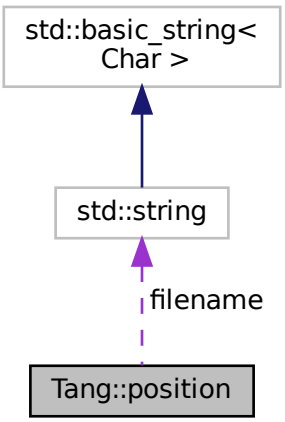
- build/generated/location.hh

5.31 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.

5.31.1 Detailed Description

A point in a source file.

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

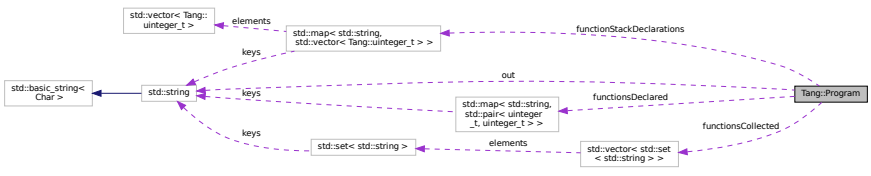
- build/generated/location.hh

5.32 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 (Tang::uinteger_t)

  Add a Tang::uinteger_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, Tang::uinteger_t jumpTarget)

  Set the target address of a Jump opcode.
- bool setFunctionStackDeclaration (size_t opcodeAddress, uinteger_t argc, uinteger_t targetPC)

  Set the stack details of a function declaration.
- void pushEnvironment (const std::shared_ptr< AstNode > &ast)

  Create a new compile/execute environment stack entry.
- void popEnvironment ()

  Remove a compile/execute environment stack entry.
- void addIdentifier (const std::string &name, std::optional< size_t > position={})

  Add an identifier to the environment.
- const std::map< std::string, size_t > & getIdentifiers () const

  Get the identifier map of the current environment.
- void addString (const std::string &name)

  Add a string to the environment.
- const std::map< std::string, size_t > & getStrings () const

  Get the string map of the current environment.

Public Attributes

- std::string out

  The output of the program, resulting from the program execution.
- std::vector< std::set< std::string > > functionsCollected

  Names of the functions that are declared in a previous or the current scope.
- std::map< std::string, std::pair< uinteger_t, uinteger_t > > functionsDeclared

  Key/value pair of the function declaration information.
- std::map< std::string, std::vector< Tang::uinteger_t > > functionStackDeclarations

  For each function name, a list of Bytecode addresses that need to be replaced by a function definition.

5.32.1 Detailed Description

Represents a compiled script or template that may be executed.

5.32.2 Member Enumeration Documentation

5.32.2.1 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.

5.32.3 Constructor & Destructor Documentation

5.32.3.1 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`.

5.32.4 Member Function Documentation

5.32.4.1 addBytecode()

```
size_t Program::addBytecode (
            Tang::uinteger_t op )
```

Add a Tang::uinteger_t to the Bytecode.

Parameters

op	The value to add to the Bytecode.

Returns

The size of the bytecode structure.

5.32.4.2 addIdentifier()

```
void Program::addIdentifier (
            const std::string & name,
            std::optional< size_t > position = {} )
```

Add an identifier to the environment.

Parameters

name	The variable to add to the environment.
position	If provided, the desired position to place the identifier.

5.32.4.3 addString()

```
void Program::addString (
            const std::string & name )
```

Add a string to the environment.

Parameters

name	The variable to add to the environment.
position	If provided, the desired position to place the identifier.

5.32.4.4 dumpBytecode()

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

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

Returns

A string containing the Opcode representation.

5.32.4.5 execute()

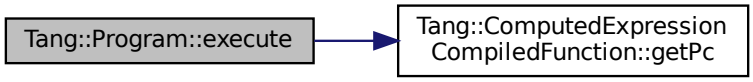
```
Program & Program::execute ( )
```

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

Returns

The current Program object.

Here is the call graph for this function:



5.32.4.6 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.

5.32.4.7 getBytecode()

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

Get the Bytecode vector.

Returns

The Bytecode vector.

5.32.4.8 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.

5.32.4.9 getIdentifiers()

```
const map< string, size_t > & Program::getIdentifiers ( ) const
```

Get the identifier map of the current environment.

Returns

A map of each identifer name to its stack position within the current environment.

5.32.4.10 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.

5.32.4.11 getStrings()

```
const map< string, size_t > & Program::getStrings ( ) const
```

Get the string map of the current environment.

Returns

A map of each identifer name to its stack position within the current environment.

5.32.4.12 pushEnvironment()

```
void Program::pushEnvironment (
            const std::shared_ptr< AstNode > & ast )
```

Create a new compile/execute environment stack entry.

Parameters

ast	The ast node from which this new environment will be formed.

5.32.4.13 setFunctionStackDeclaration()

```
bool Program::setFunctionStackDeclaration (
            size_t opcodeAddress,
            uinteger_t argc,
            uinteger_t targetPC )
```

Set the stack details of a function declaration.

Parameters

opcodeAddress	The location of the FUNCTION opcode.
argc	The argument count to set.
targetPC	The bytecode address of the start of the function.

5.32.4.14 setJumpTarget()

```
bool Program::setJumpTarget (
            size_t opcodeAddress,
            Tang::uinteger_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.

5.32.5 Member Data Documentation

5.32.5.1 functionsDeclared

```
std::map<std::string, std::pair<uinteger_t, uinteger_t> > Tang::Program::functionsDeclared
```

Key/value pair of the function declaration information.

The key is the name of the function. The value is a `pair` of the `argc` value and the `targetPC` value.

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

5.33 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.

5.33.1 Detailed Description

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

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

5.33.2 Member Function Documentation

5.33.2.1 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.

5.33.2.2 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.

5.33.2.3 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

5.34 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.

5.34.1 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).

5.34.2 Constructor & Destructor Documentation

5.34.2.1 TangBase()

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

The constructor.

Isn't it glorious.

5.34.3 Member Function Documentation

5.34.3.1 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:

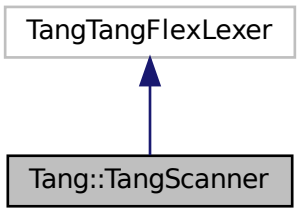
- include/tangBase.hpp
- src/tangBase.cpp

5.35 Tang::TangScanner Class Reference

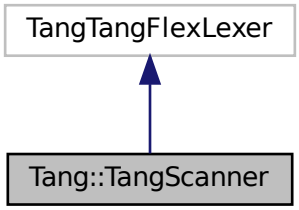
The Flex lexer class for the main Tang language.

`#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.

5.35.1 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.

5.35.2 Constructor & Destructor Documentation

5.35.2.1 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)

5.35.3 Member Function Documentation

5.35.3.1 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

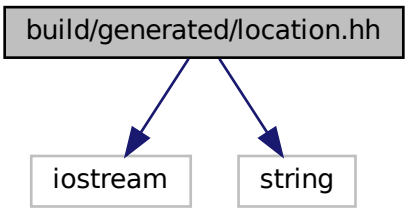
File Documentation

6.1 build/generated/location.hh File Reference

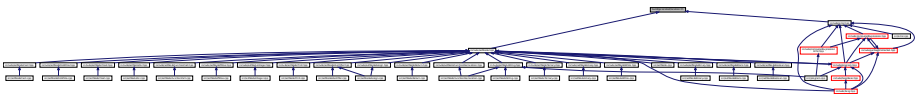
Define the Tang ::location class.

```
#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

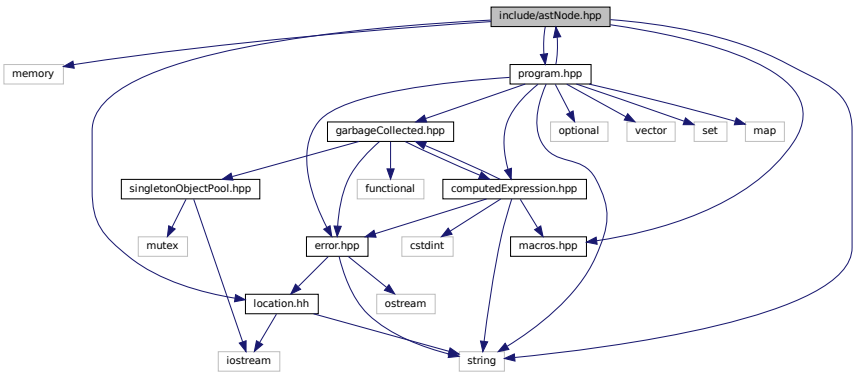
ostr	the destination output stream
pos	a reference to the position to redirect

6.2 include/astNode.hpp File Reference

Declare the Tang::AstNode base class.

```
#include <memory>
#include <string>
#include "location.hh"
#include "macros.hpp"
#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).

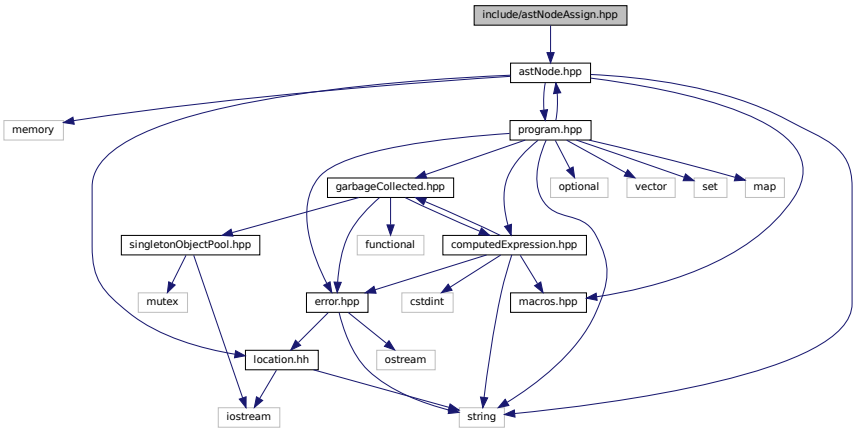
6.2.1 Detailed Description

Declare the Tang::AstNode base class.

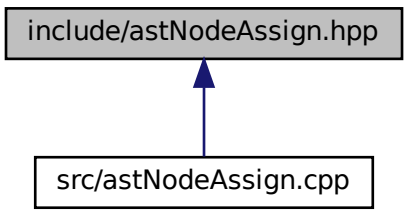
6.3 include/astNodeAssign.hpp File Reference

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

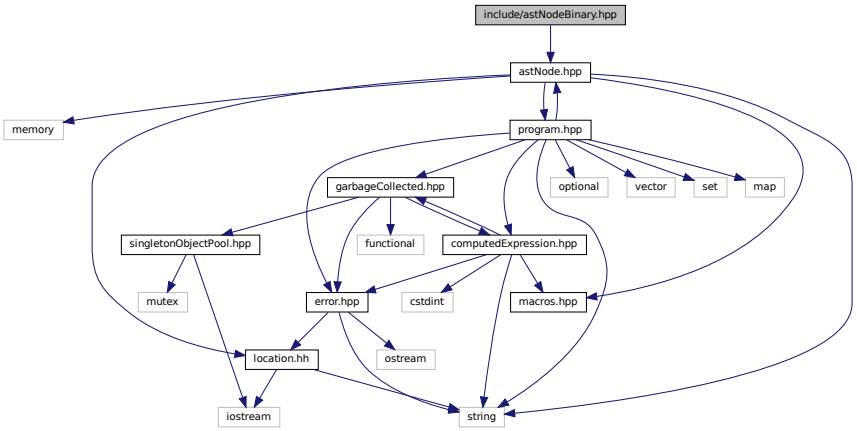
Declare the Tang::AstNodeAssign class.

6.4 include/astNodeBinary.hpp File Reference

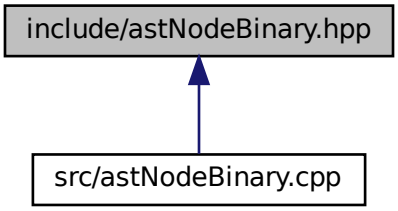
Declare the Tang::AstNodeBinary class.

```
#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

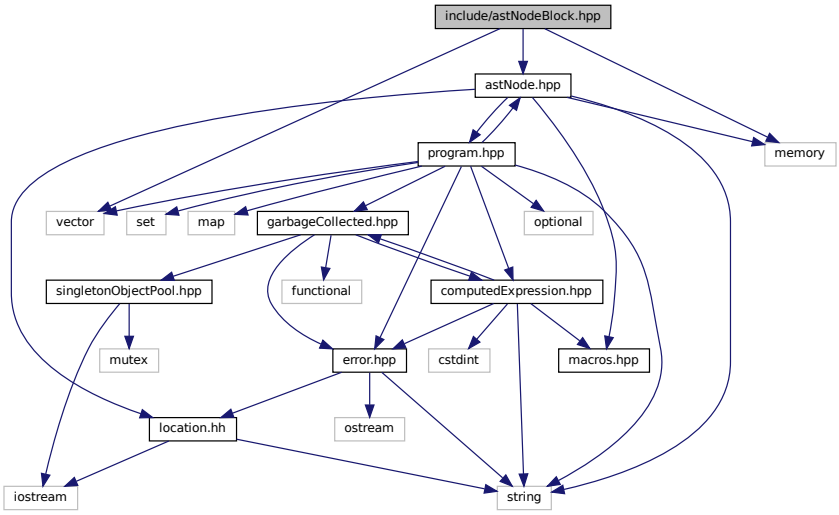
Declare the Tang::AstNodeBinary class.

6.5 include/astNodeBlock.hpp File Reference

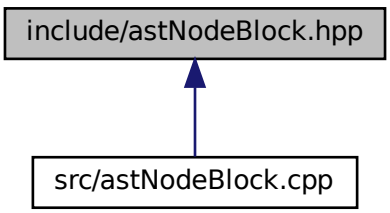
Declare the Tang::AstNodeBlock class.

```
#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

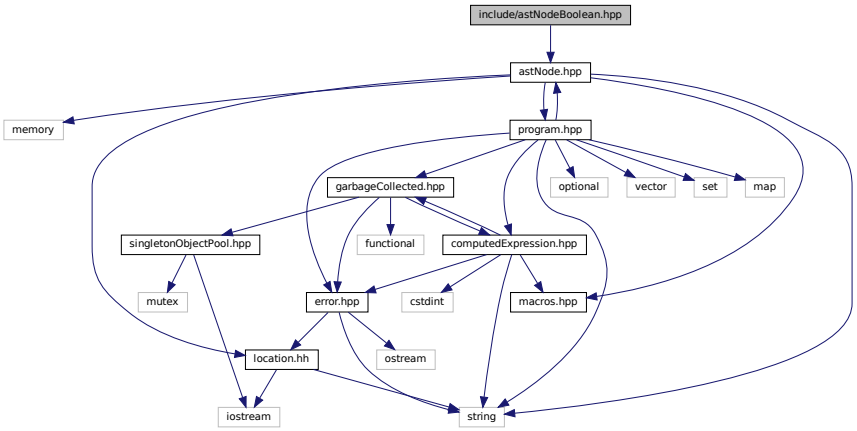
Declare the Tang::AstNodeBlock class.

6.6 include/astNodeBoolean.hpp File Reference

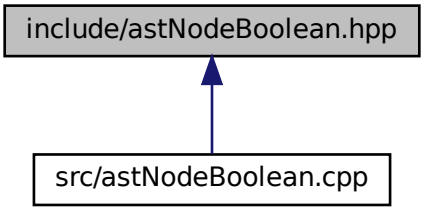
Declare the Tang::AstNodeBoolean class.

```
#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

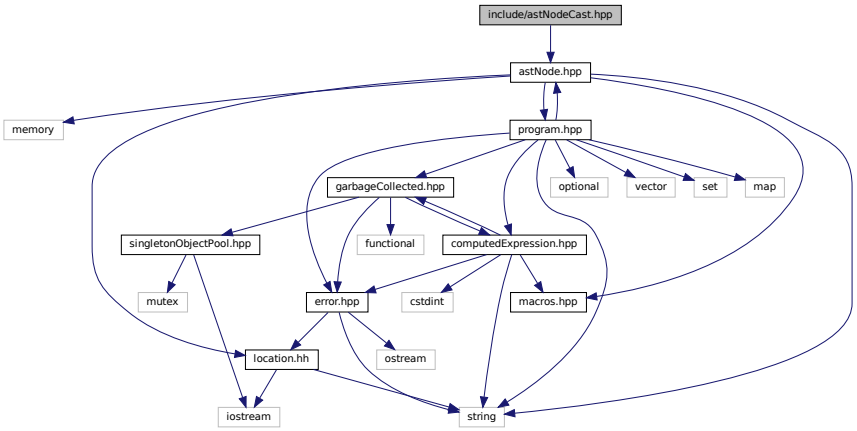
Declare the Tang::AstNodeBoolean class.

6.7 include/astNodeCast.hpp File Reference

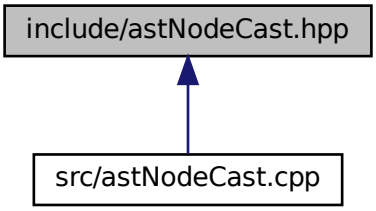
Declare the Tang::AstNodeCast class.

```
#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

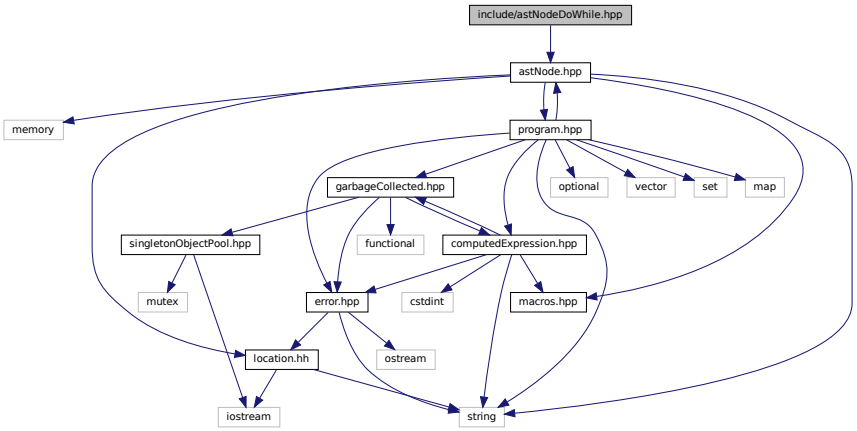
Declare the Tang::AstNodeCast class.

6.8 include/astNodeDoWhile.hpp File Reference

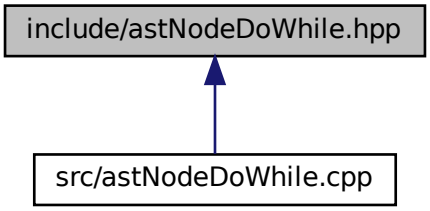
Declare the Tang::AstNodeDoWhile class.

```
#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

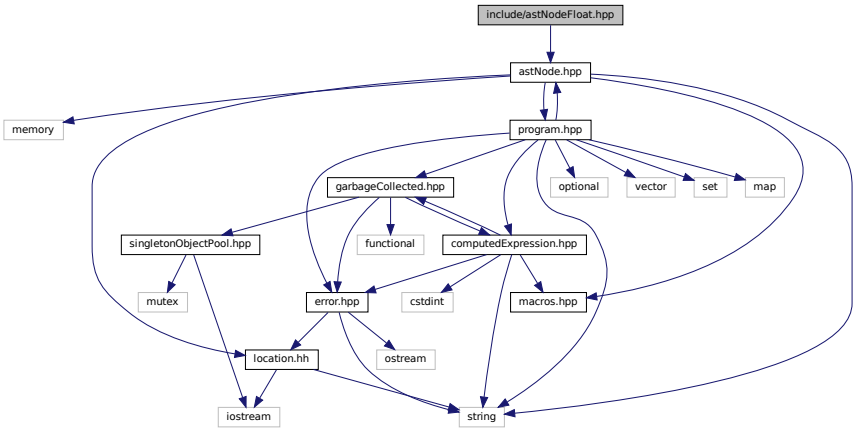
Declare the Tang::AstNodeDoWhile class.

6.9 include/astNodeFloat.hpp File Reference

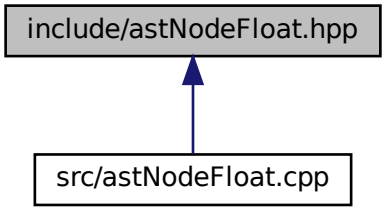
Declare the Tang::AstNodeFloat class.

```
#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

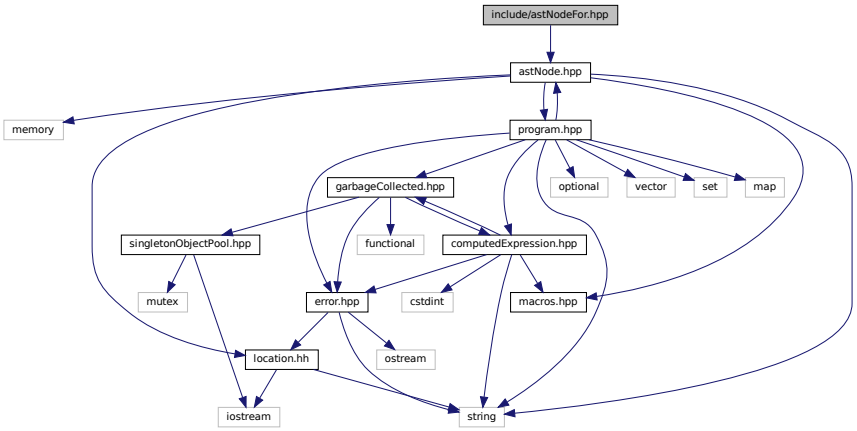
Declare the Tang::AstNodeFloat class.

6.10 include/astNodeFor.hpp File Reference

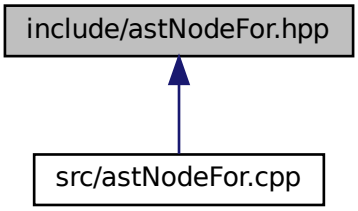
Declare the Tang::AstNodeFor class.

```
#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

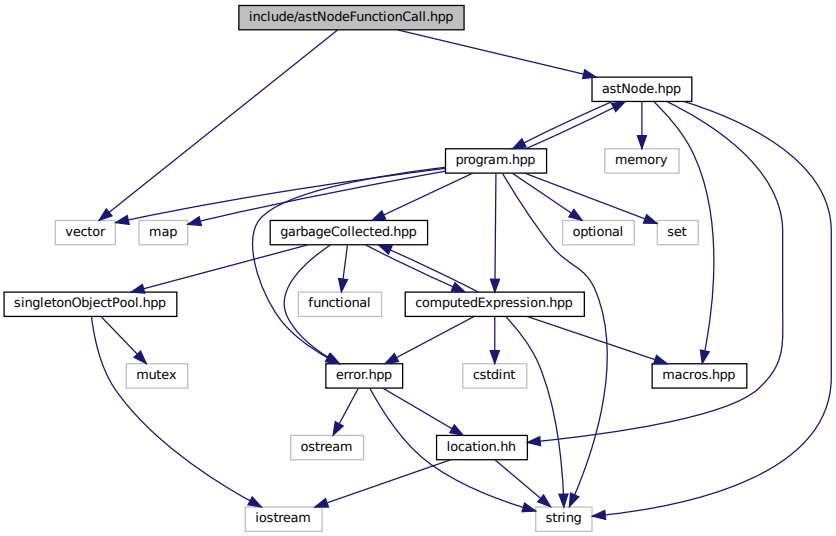
Declare the Tang::AstNodeFor class.

6.11 include/astNodeFunctionCall.hpp File Reference

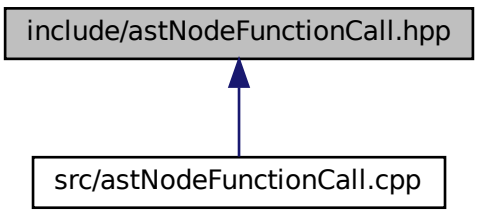
Declare the Tang::AstNodeFunctionCall class.

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

Include dependency graph for astNodeFunctionCall.hpp:



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



Classes

- class Tang::AstNodeFunctionCall

  An AstNode that represents a function call.

6.11.1 Detailed Description

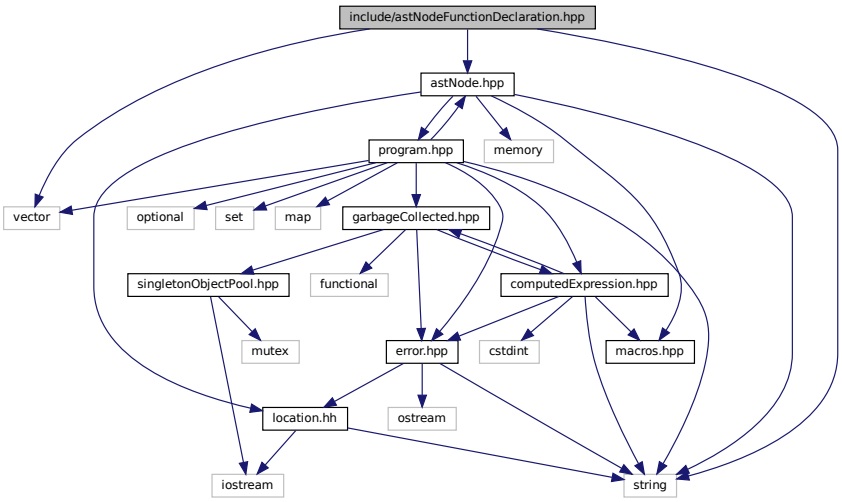
Declare the Tang::AstNodeFunctionCall class.

6.12 include/astNodeFunctionDeclaration.hpp File Reference

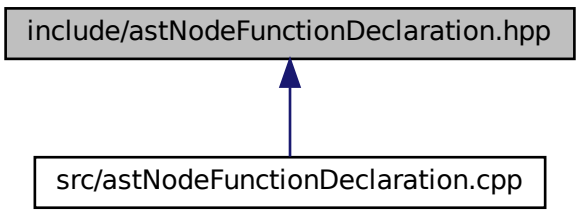
Declare the Tang::AstNodeFunctionDeclaration class.

```
#include <string>
#include <vector>
#include "astNode.hpp"
```

Include dependency graph for astNodeFunctionDeclaration.hpp:



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



Classes

- class Tang::AstNodeFunctionDeclaration

  An AstNode that represents a function declaration.

6.12.1 Detailed Description

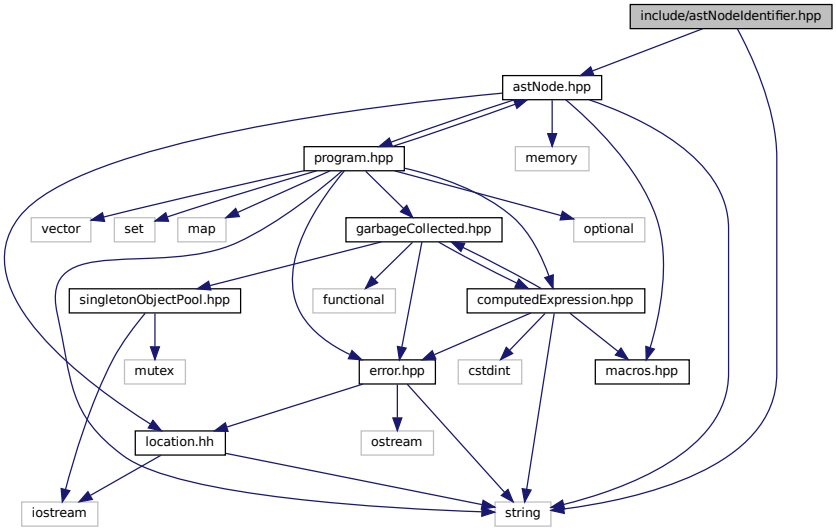
Declare the Tang::AstNodeFunctionDeclaration class.

6.13 include/astNodeIdentifier.hpp File Reference

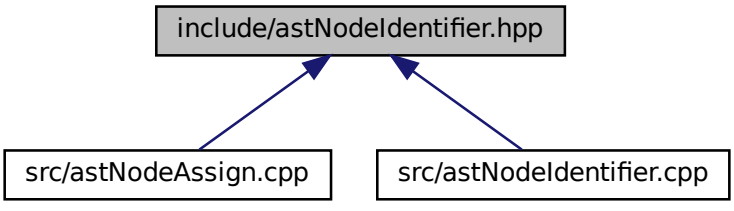
Declare the Tang::AstNodeIdentifier class.

```
#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.

6.13.1 Detailed Description

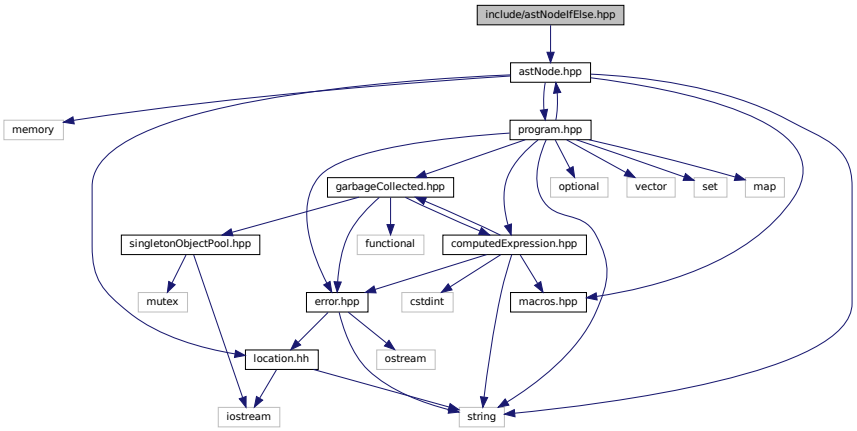
Declare the Tang::AstNodeIdentifier class.

6.14 include/astNodeIfElse.hpp File Reference

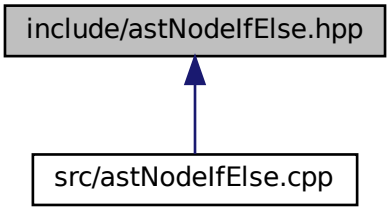
Declare the Tang::AstNodeIfElse class.

```
#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.

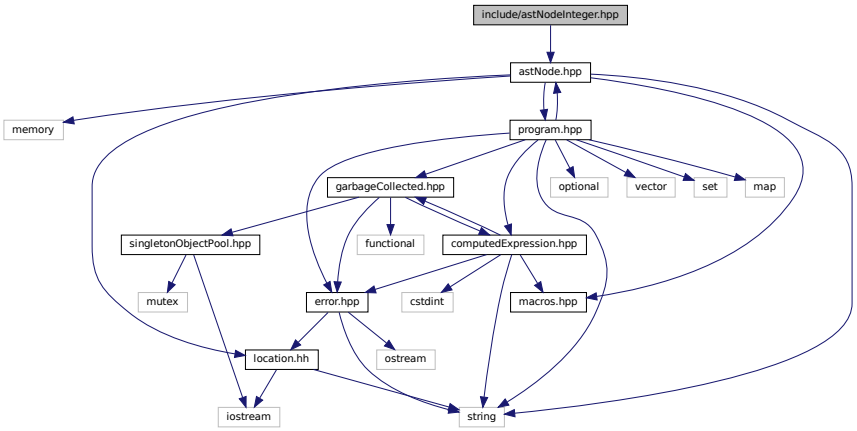
6.14.1 Detailed Description

Declare the Tang::AstNodeIfElse class.

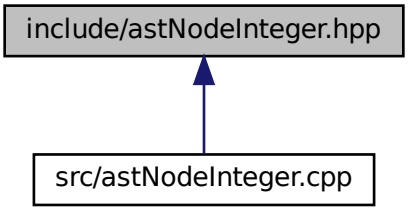
6.15 include/astNodeInteger.hpp File Reference

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.

6.15.1 Detailed Description

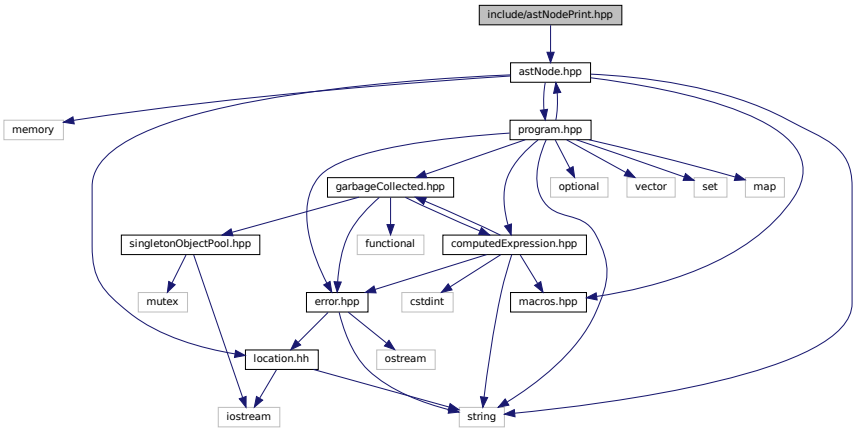
Declare the Tang::AstNodeInteger class.

6.16 include/astNodePrint.hpp File Reference

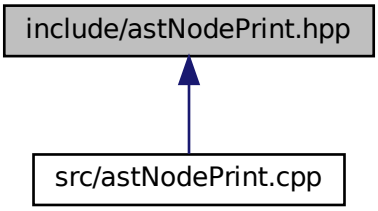
Declare the Tang::AstNodePrint class.

```
#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.

6.16.1 Detailed Description

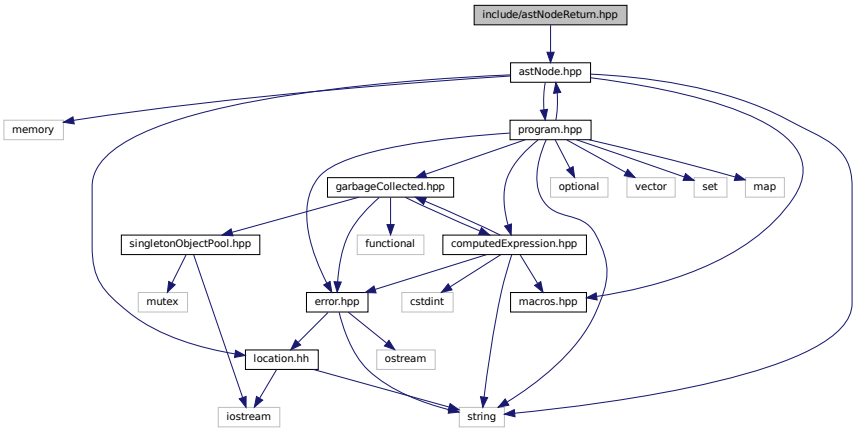
Declare the Tang::AstNodePrint class.

6.17 include/astNodeReturn.hpp File Reference

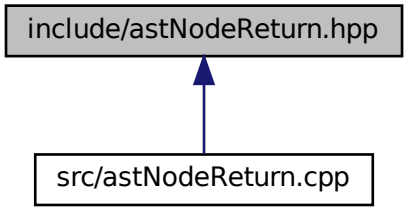
Declare the Tang::AstNodeReturn class.

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

Include dependency graph for astNodeReturn.hpp:



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



Classes

- class Tang::AstNodeReturn

  An AstNode that represents a `return` statement.

6.17.1 Detailed Description

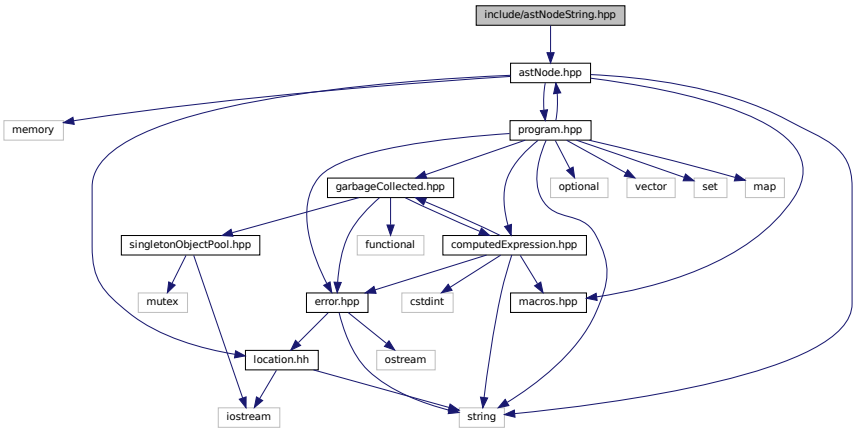
Declare the Tang::AstNodeReturn class.

6.18 include/astNodeString.hpp File Reference

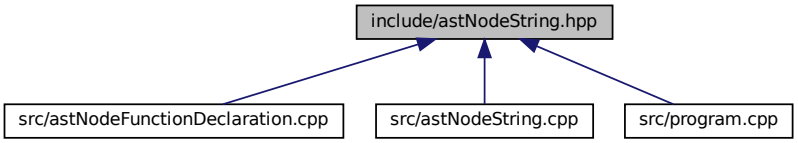
Declare the Tang::AstNodeString class.

```
#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.

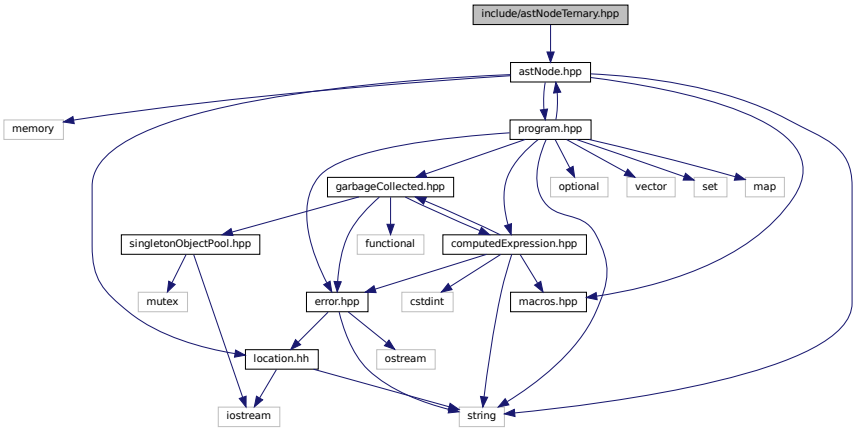
6.18.1 Detailed Description

Declare the Tang::AstNodeString class.

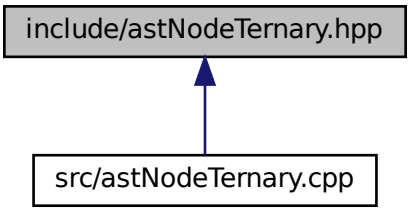
6.19 include/astNodeTernary.hpp File Reference

Declare the Tang::AstNodeTernary class.

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



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



Classes

- class Tang::AstNodeTernary

  An AstNode that represents a ternary expression.

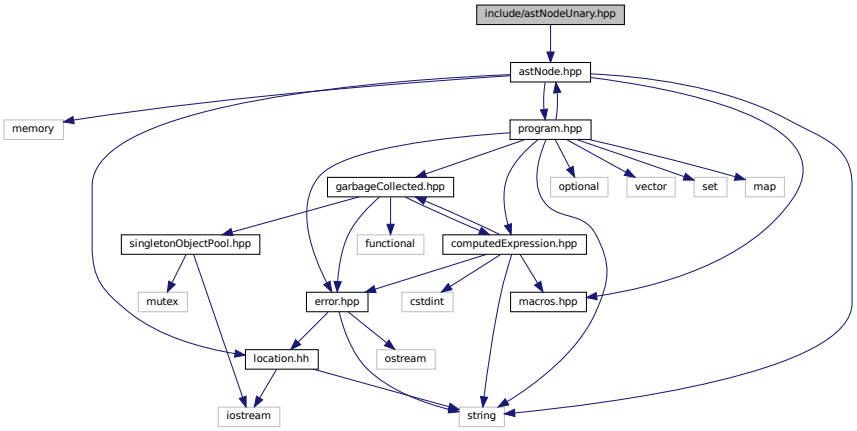
6.19.1 Detailed Description

Declare the Tang::AstNodeTernary class.

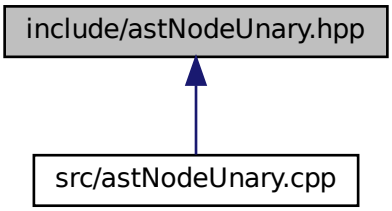
6.20 include/astNodeUnary.hpp File Reference

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.

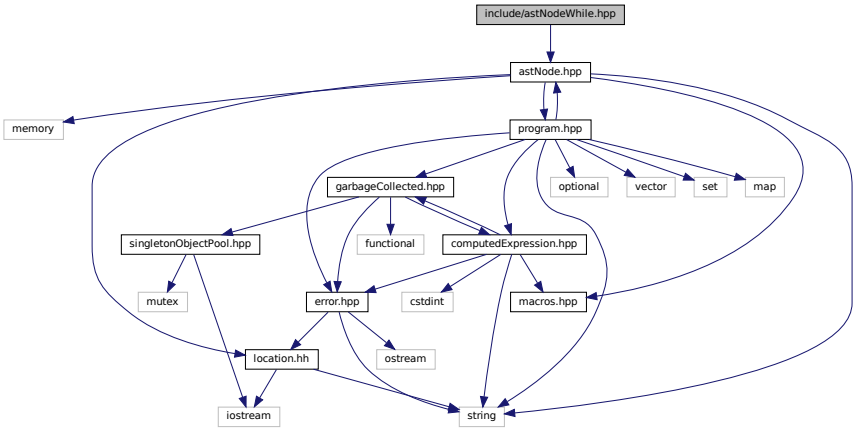
6.20.1 Detailed Description

Declare the Tang::AstNodeUnary class.

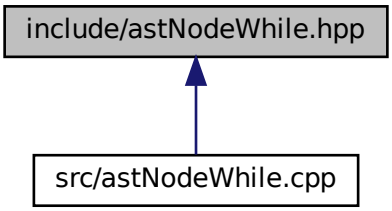
6.21 include/astNodeWhile.hpp File Reference

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.

6.21.1 Detailed Description

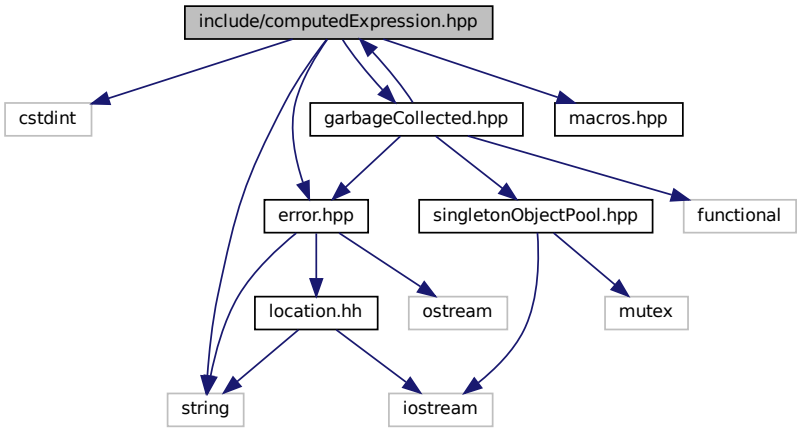
Declare the Tang::AstNodeWhile class.

6.22 include/computedExpression.hpp File Reference

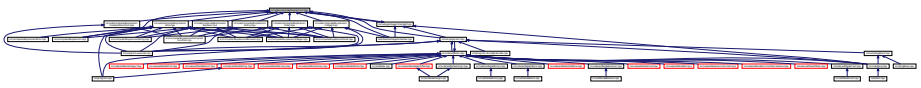
Declare the Tang::ComputedExpression base class.

```
#include <cstdint>
#include <string>
#include "macros.hpp"
#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.

6.22.1 Detailed Description

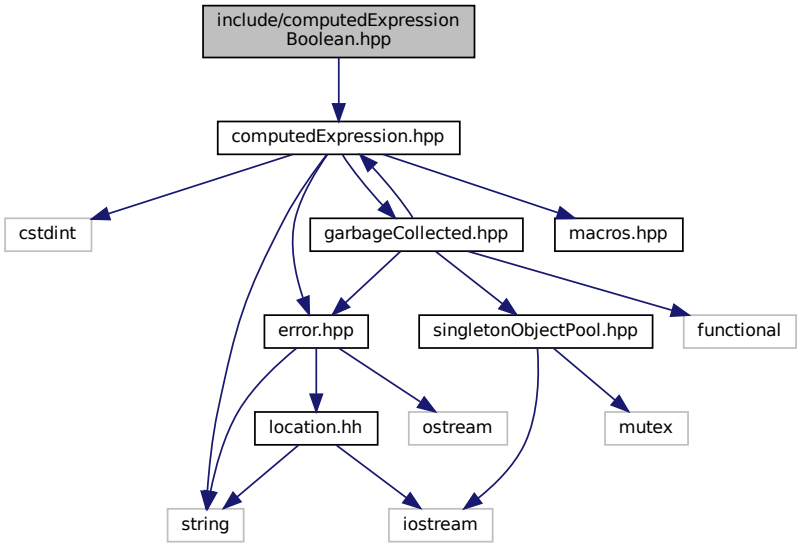
Declare the Tang::ComputedExpression base class.

6.23 include/computedExpressionBoolean.hpp File Reference

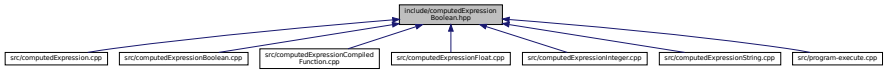
Declare the Tang::ComputedExpressionBoolean class.

```
#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.

6.23.1 Detailed Description

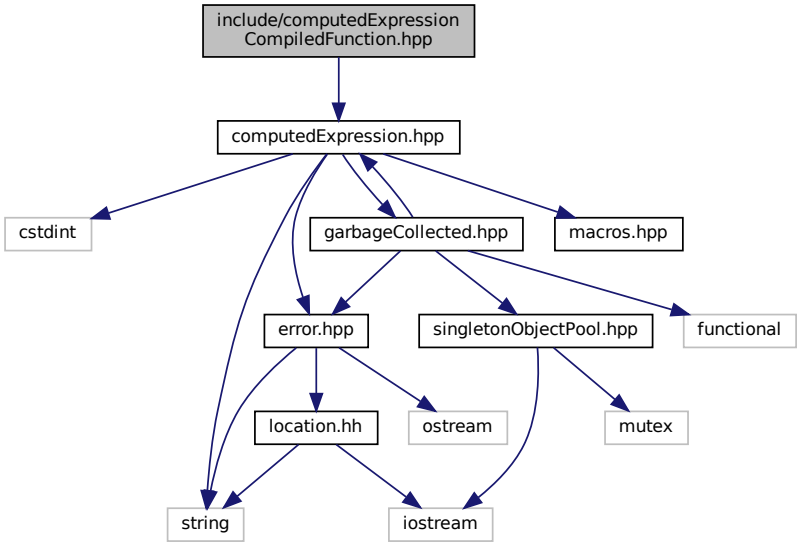
Declare the Tang::ComputedExpressionBoolean class.

6.24 include/computedExpressionCompiledFunction.hpp File Reference

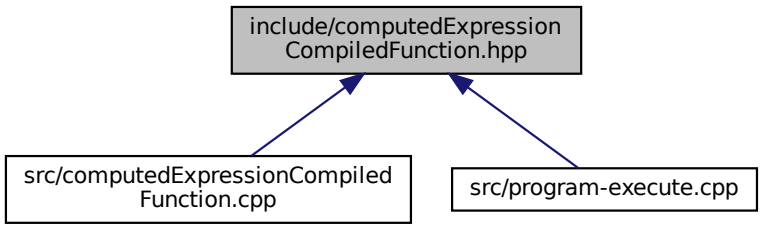
Declare the Tang::ComputedExpressionCompiledFunction class.

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

Include dependency graph for computedExpressionCompiledFunction.hpp:



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



Classes

- class Tang::ComputedExpressionCompiledFunction

  Represents a Compiled Function declared in the script.

6.24.1 Detailed Description

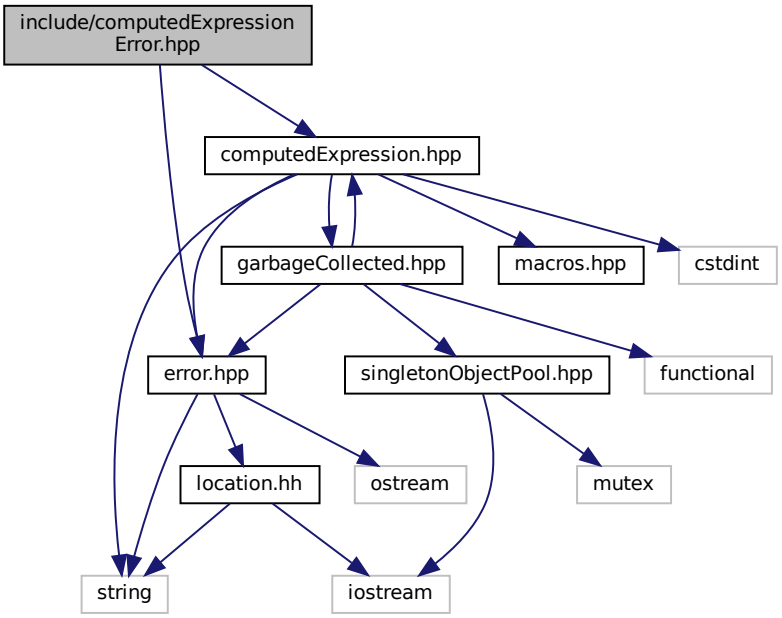
Declare the Tang::ComputedExpressionCompiledFunction class.

6.25 include/computedExpressionError.hpp File Reference

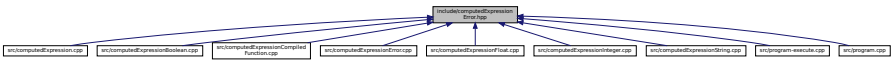
Declare the Tang::ComputedExpressionError class.

```
#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.

6.25.1 Detailed Description

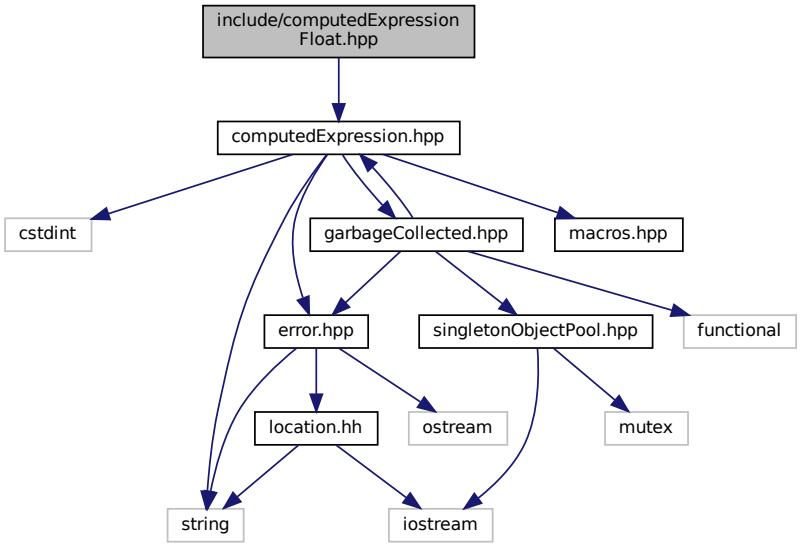
Declare the Tang::ComputedExpressionError class.

6.26 include/computedExpressionFloat.hpp File Reference

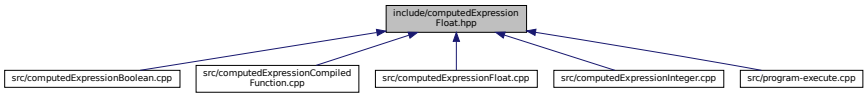
Declare the Tang::ComputedExpressionFloat class.

```
#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.

6.26.1 Detailed Description

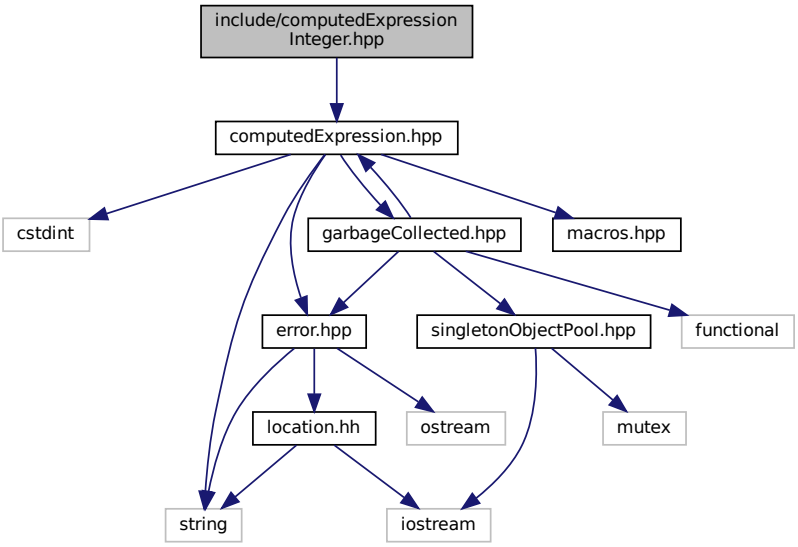
Declare the Tang::ComputedExpressionFloat class.

6.27 include/computedExpressionInteger.hpp File Reference

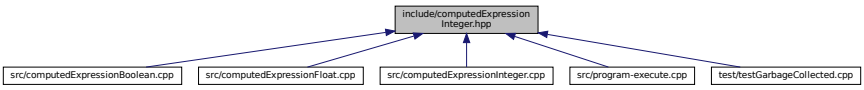
Declare the Tang::ComputedExpressionInteger class.

```
#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.

6.27.1 Detailed Description

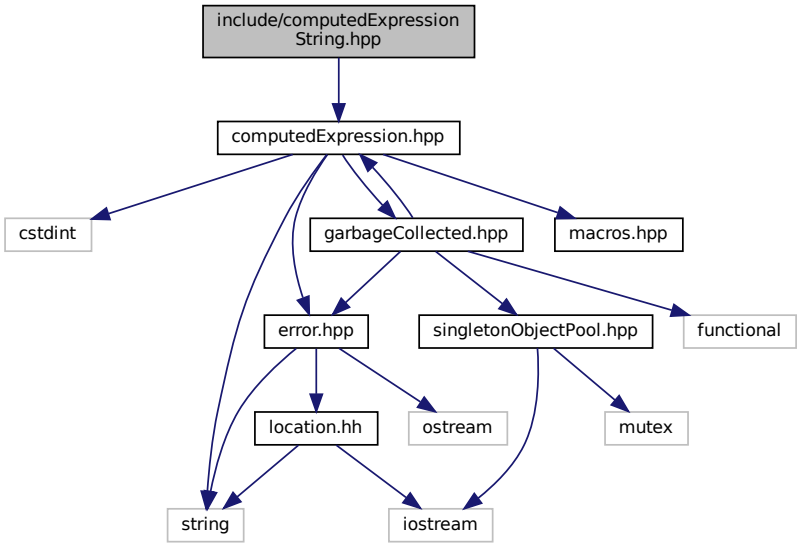
Declare the Tang::ComputedExpressionInteger class.

6.28 include/computedExpressionString.hpp File Reference

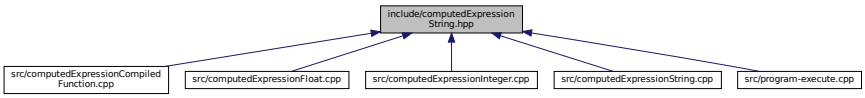
Declare the Tang::ComputedExpressionString class.

```
#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.

6.28.1 Detailed Description

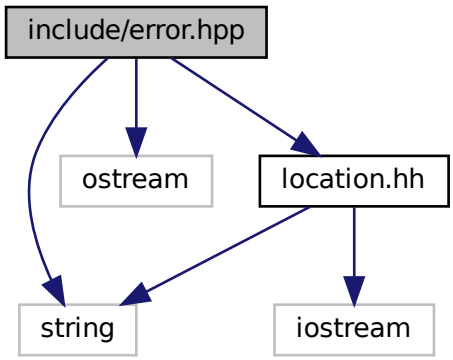
Declare the Tang::ComputedExpressionString class.

6.29 include/error.hpp File Reference

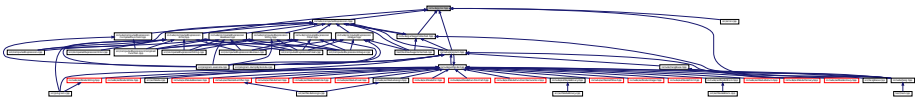
Declare the Tang::Error class used to describe syntax and runtime errors.

```
#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.

6.29.1 Detailed Description

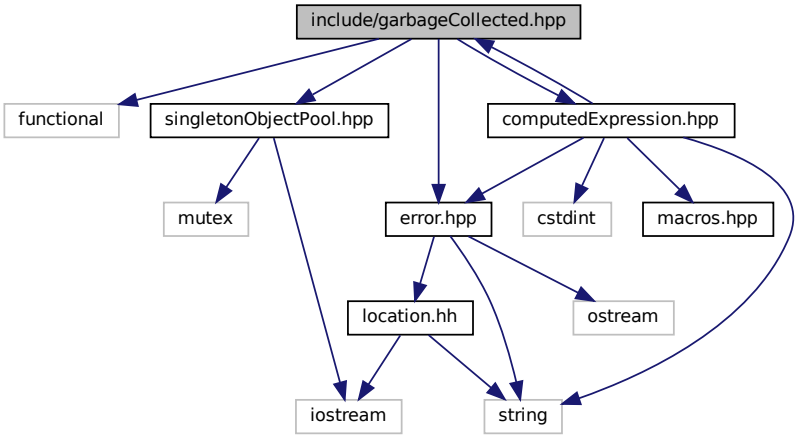
Declare the Tang::Error class used to describe syntax and runtime errors.

6.30 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.

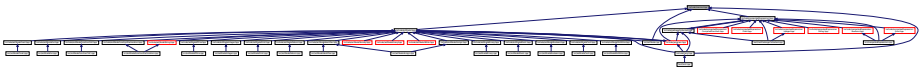
6.30.1 Detailed Description

Declare the Tang::GarbageCollected class.

6.31 include/macros.hpp File Reference

Contains generic macros.

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



Typedefs

- using Tang::integer_t = int32_t

  Define the size of signed integers used by Tang.
- using Tang::uinteger_t = int32_t

  Define the size of integers used by Tang.
- using Tang::float_t = float

  Define the size of floats used by Tang.

6.31.1 Detailed Description

Contains generic macros.

6.32 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 , FUNCTION , ADD , SUBTRACT ,
  MULTIPLY , DIVIDE , MODULO , NEGATIVE ,
  NOT , LT , LTE , GT ,
  GTE , EQ , NEQ , CASTINTEGER ,
  CASTFLOAT , CASTBOOLEAN , CALLFUNC , RETURN ,
  PRINT }

6.32.1 Detailed Description

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

6.32.2 Enumeration Type Documentation

6.32.2.1 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.
FUNCTION	Get argc, PC#: push function(argc, PC #)
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.
CALLFUNC	Get argc, Pop a function, execute function if argc matches.
RETURN	Get stack #, pop return val, pop (stack #) times, push val, restore fp, restore pc.
PRINT	Pop val, print(val), push error or NULL.

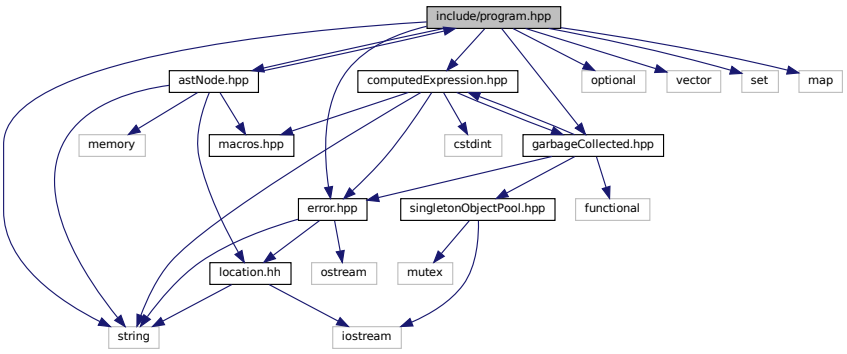
6.33 include/program.hpp File Reference

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

```
#include <string>
#include <optional>
#include <vector>
#include <set>
#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< Tang::uinteger_t >

  Contains the Opcodes of a compiled program.

6.33.1 Detailed Description

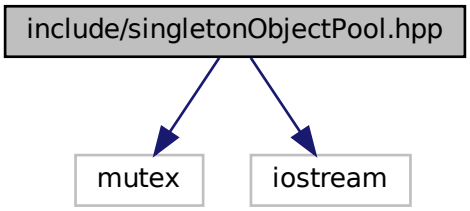
Declare the Tang::Program class used to compile and execute source code.

6.34 include/singletonObjectPool.hpp File Reference

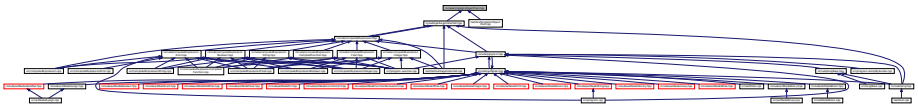
Declare the Tang::SingletonObjectPool class.

```
#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.

6.34.1 Detailed Description

Declare the Tang::SingletonObjectPool class.

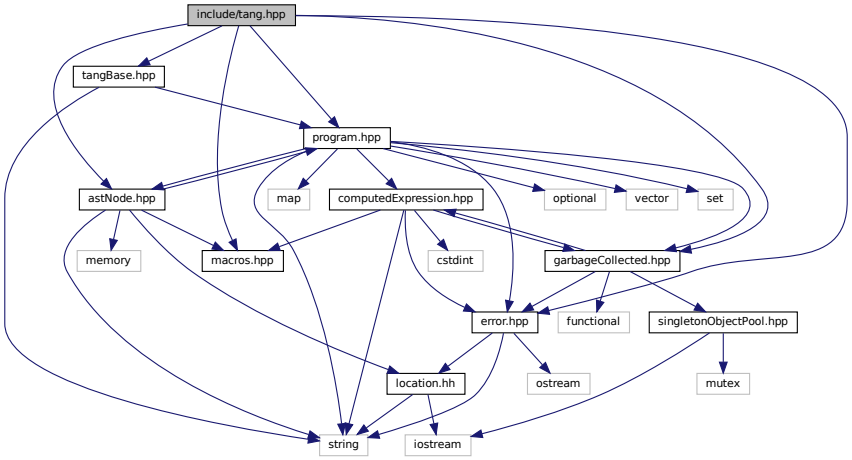
6.35 include/tang.hpp File Reference

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

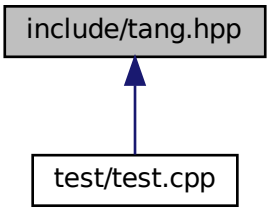
```
#include "macros.hpp"
#include "tangBase.hpp"
#include "astNode.hpp"
#include "error.hpp"
#include "garbageCollected.hpp"
```

```
#include "program.hpp"
```

Include dependency graph for tang.hpp:



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



6.35.1 Detailed Description

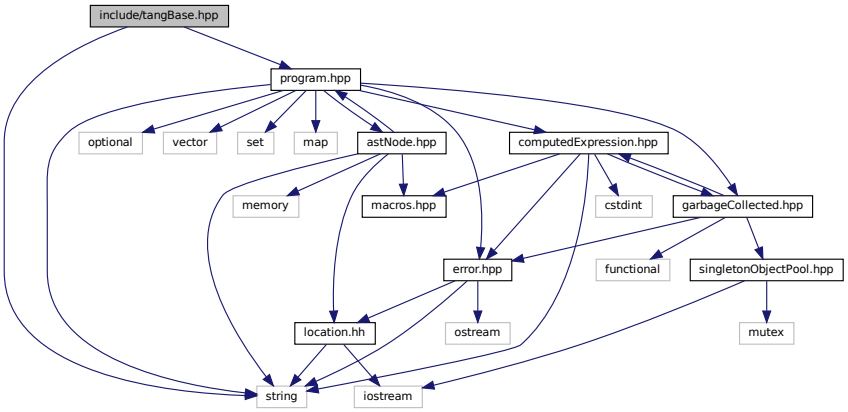
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

6.36 include/tangBase.hpp File Reference

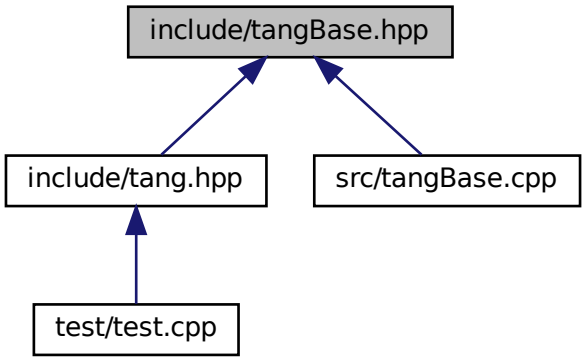
Declare the Tang::TangBase class used to interact with Tang.

```
#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.

6.36.1 Detailed Description

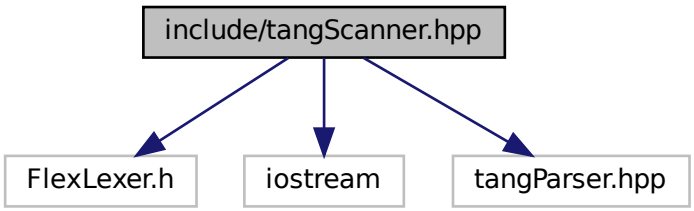
Declare the Tang::TangBase class used to interact with Tang.

6.37 include/tangScanner.hpp File Reference

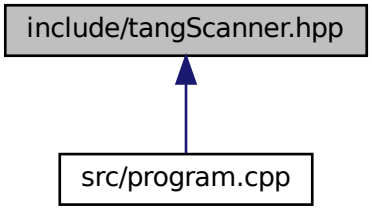
Declare the Tang::TangScanner used to tokenize a Tang script.

```
#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()

6.37.1 Detailed Description

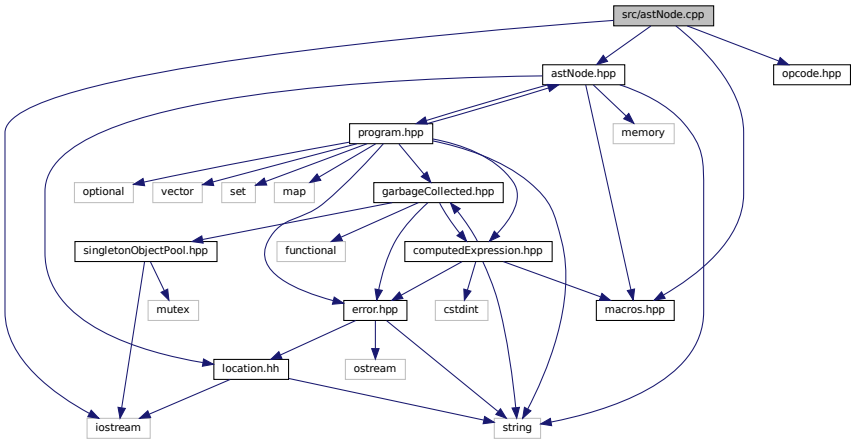
Declare the Tang::TangScanner used to tokenize a Tang script.

6.38 src/astNode.cpp File Reference

Define the Tang::AstNode class.

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

Include dependency graph for astNode.cpp:



6.38.1 Detailed Description

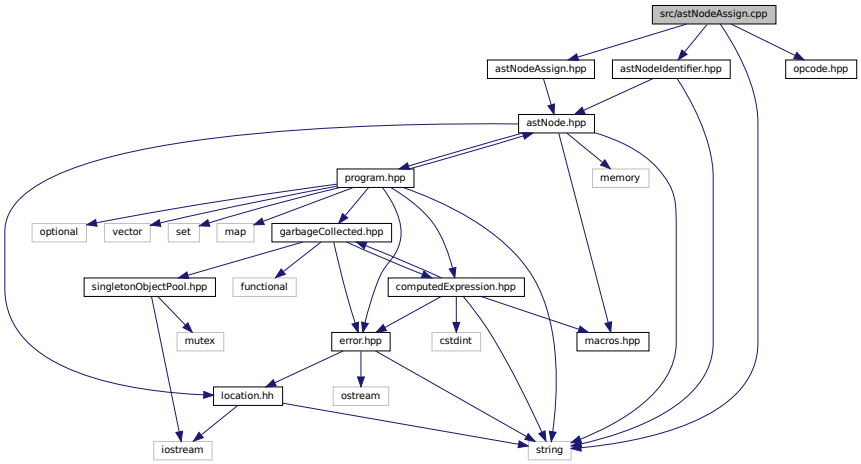
Define the Tang::AstNode class.

6.39 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:



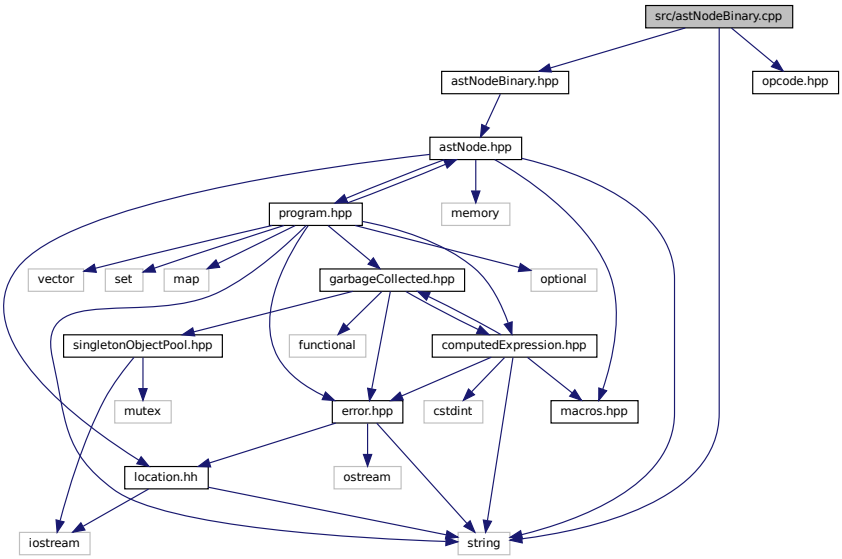
6.39.1 Detailed Description

Define the Tang::AstNodeAssign class.

6.40 src/astNodeBinary.cpp File Reference

Define the Tang::AstNodeBinary class.

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



6.40.1 Detailed Description

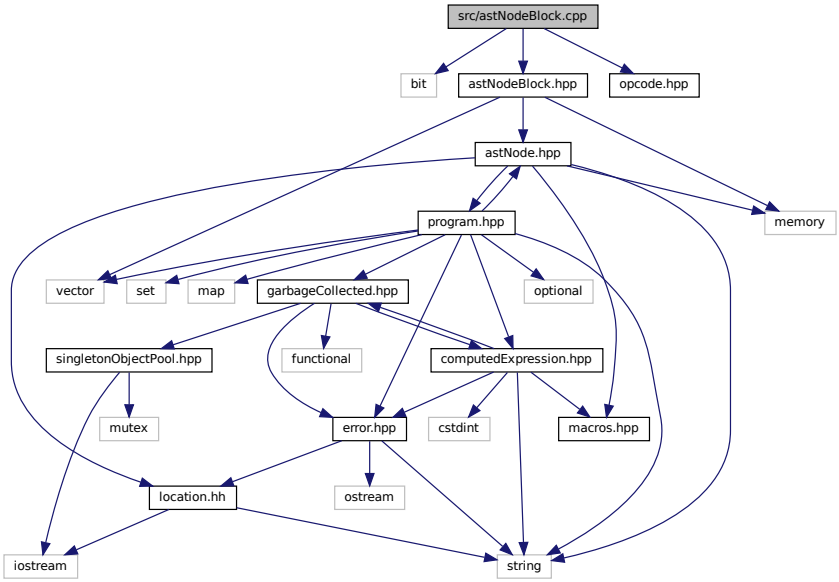
Define the Tang::AstNodeBinary class.

6.41 src/astNodeBlock.cpp File Reference

Define the Tang::AstNodeBlock class.

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

Include dependency graph for astNodeBlock.cpp:



6.41.1 Detailed Description

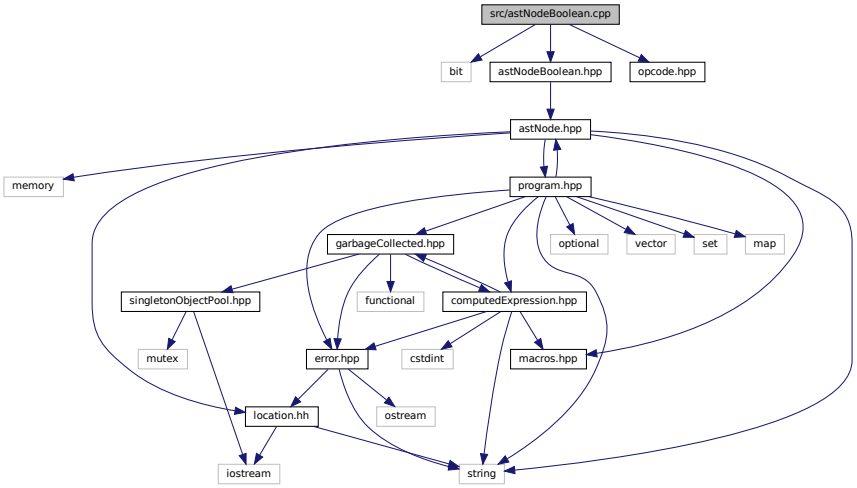
Define the Tang::AstNodeBlock class.

6.42 src/astNodeBoolean.cpp File Reference

Define the Tang::AstNodeBoolean class.

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

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



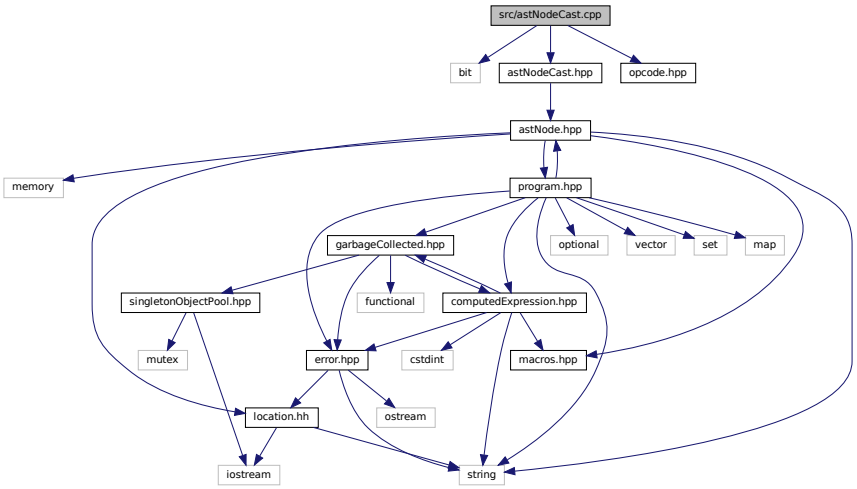
6.42.1 Detailed Description

Define the Tang::AstNodeBoolean class.

6.43 src/astNodeCast.cpp File Reference

Define the Tang::AstNodeCast class.

```
#include <bit>
#include "astNodeCast.hpp"
#include "opcode.hpp"
```
Include dependency graph for astNodeCast.cpp:



6.43.1 Detailed Description

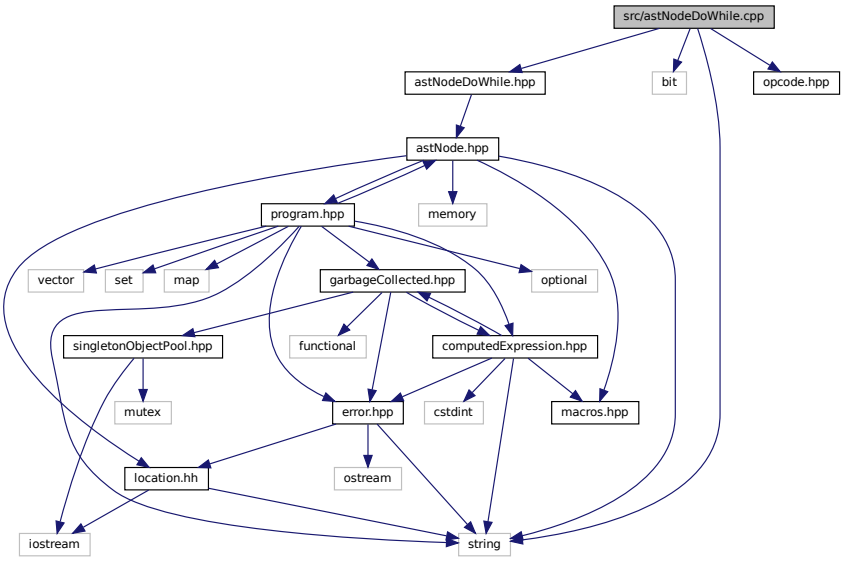
Define the Tang::AstNodeCast class.

6.44 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:



6.44.1 Detailed Description

Define the Tang::AstNodeDoWhile class.

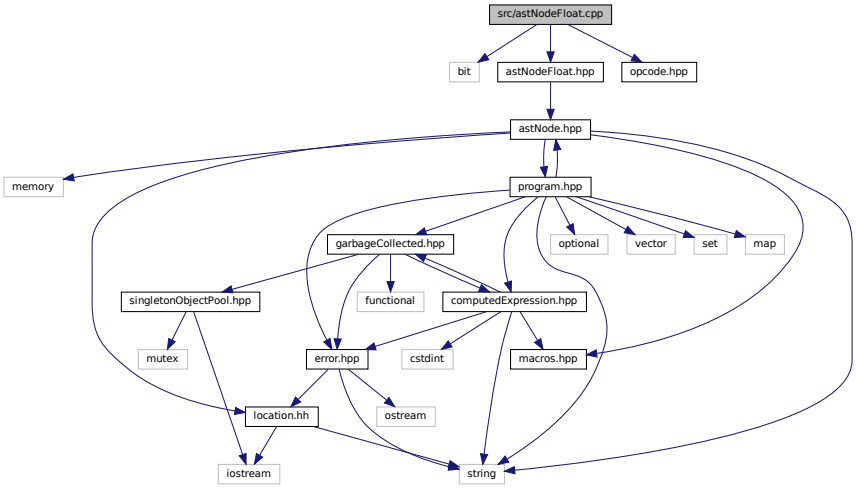
6.45 src/astNodeFloat.cpp File Reference

Define the Tang::AstNodeFloat class.

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

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

Include dependency graph for astNodeFloat.cpp:



6.45.1 Detailed Description

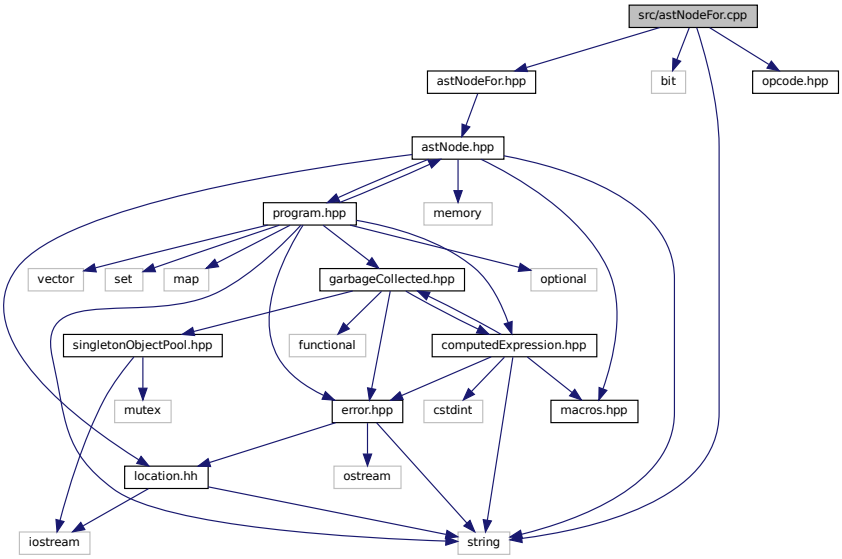
Define the Tang::AstNodeFloat class.

6.46 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:



6.46.1 Detailed Description

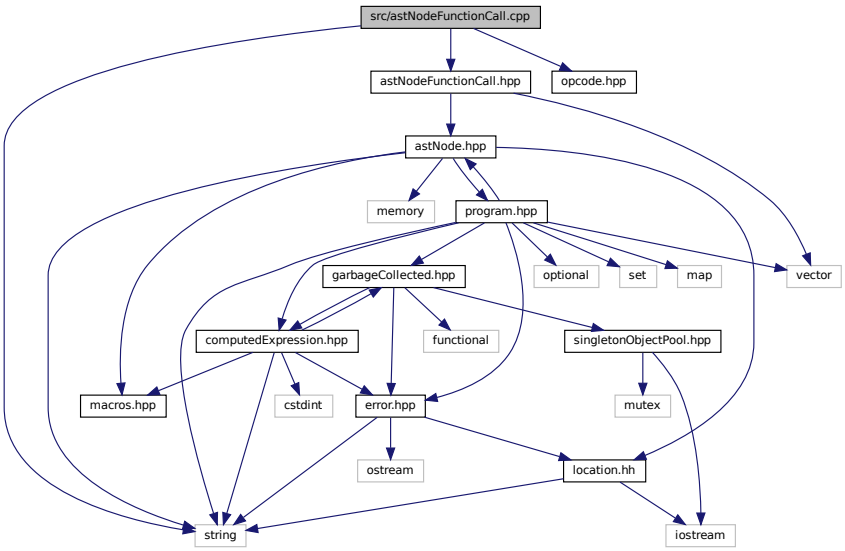
Define the Tang::AstNodeFor class.

6.47 src/astNodeFunctionCall.cpp File Reference

Define the Tang::AstNodeFunctionCall class.

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

Include dependency graph for astNodeFunctionCall.cpp:



6.47.1 Detailed Description

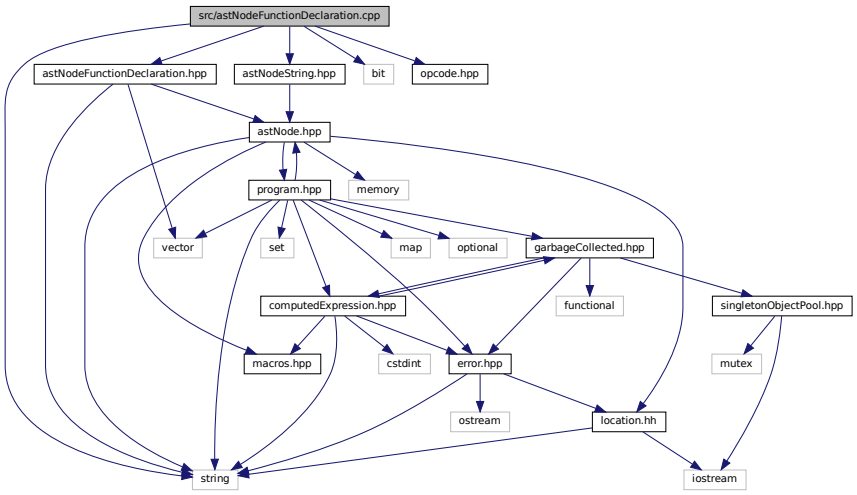
Define the Tang::AstNodeFunctionCall class.

6.48 src/astNodeFunctionDeclaration.cpp File Reference

Define the Tang::AstNodeFunctionDeclaration class.

```
#include <string>
#include <bit>
#include "astNodeFunctionDeclaration.hpp"
#include "astNodeString.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeFunctionDeclaration.cpp:



6.48.1 Detailed Description

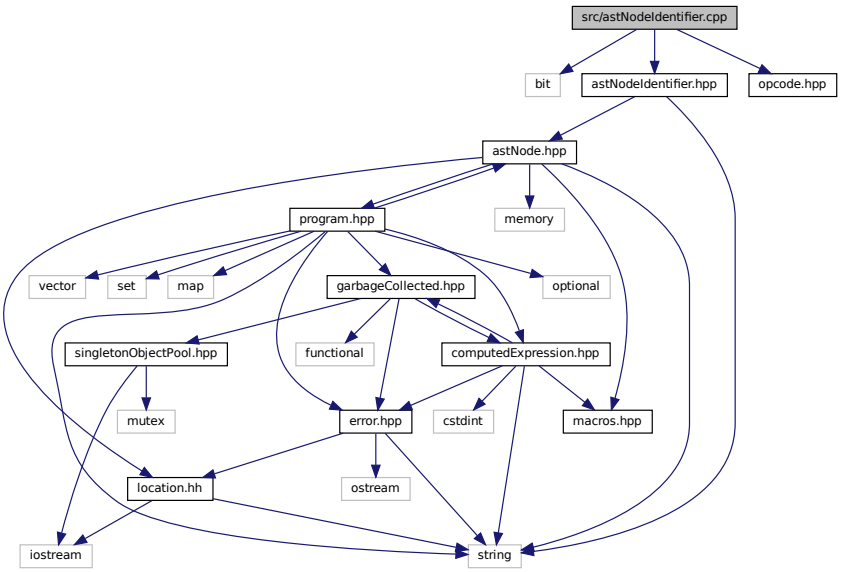
Define the Tang::AstNodeFunctionDeclaration class.

6.49 src/astNodeIdentifier.cpp File Reference

Define the Tang::AstNodeIdentifier class.

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

Include dependency graph for astNodeIdentifier.cpp:



6.49.1 Detailed Description

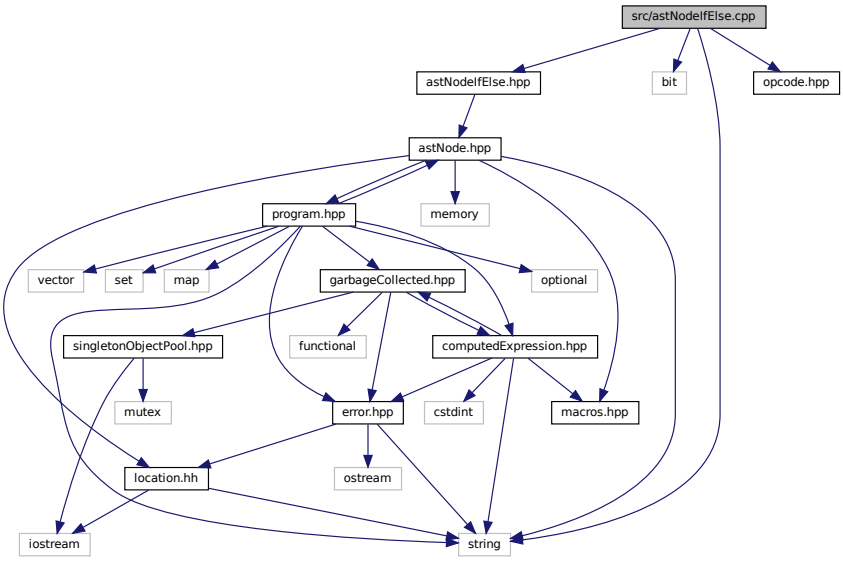
Define the Tang::AstNodeIdentifier class.

6.50 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:



6.50.1 Detailed Description

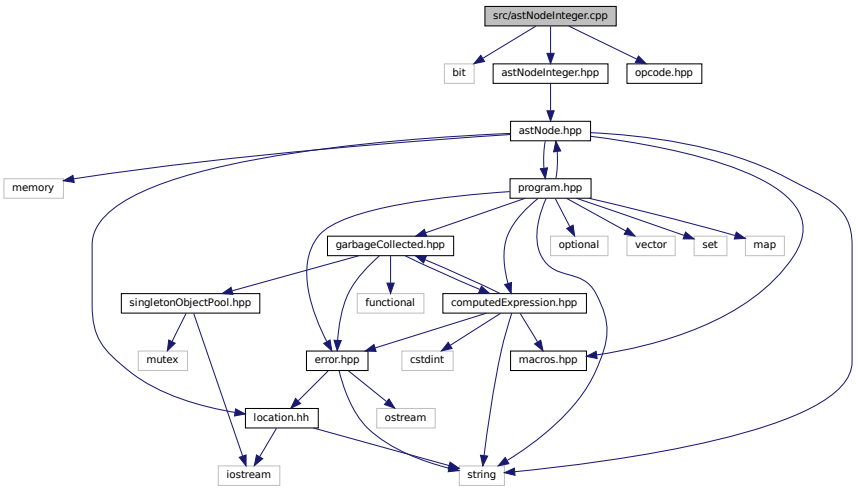
Define the Tang::AstNodeIfElse class.

6.51 src/astNodeInteger.cpp File Reference

Define the Tang::AstNodeInteger class.

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

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



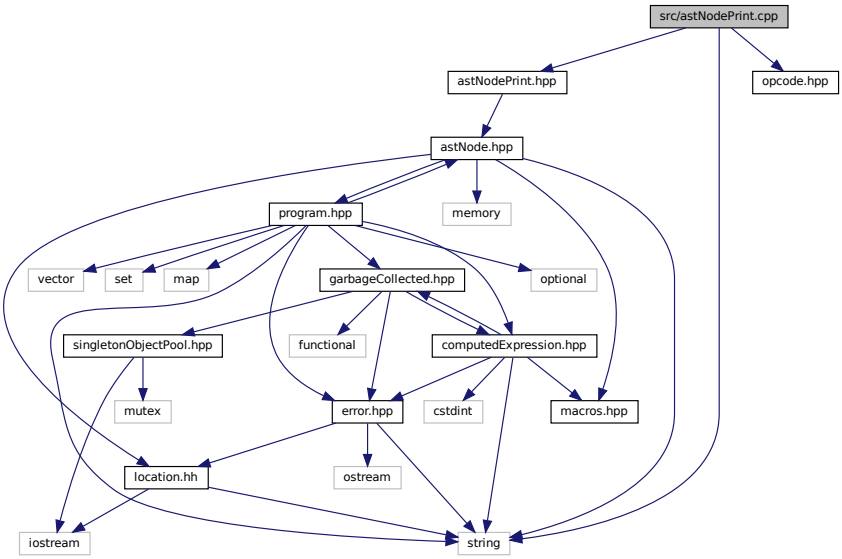
6.51.1 Detailed Description

Define the Tang::AstNodeInteger class.

6.52 src/astNodePrint.cpp File Reference

Define the Tang::AstNodePrint class.

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



6.52.1 Detailed Description

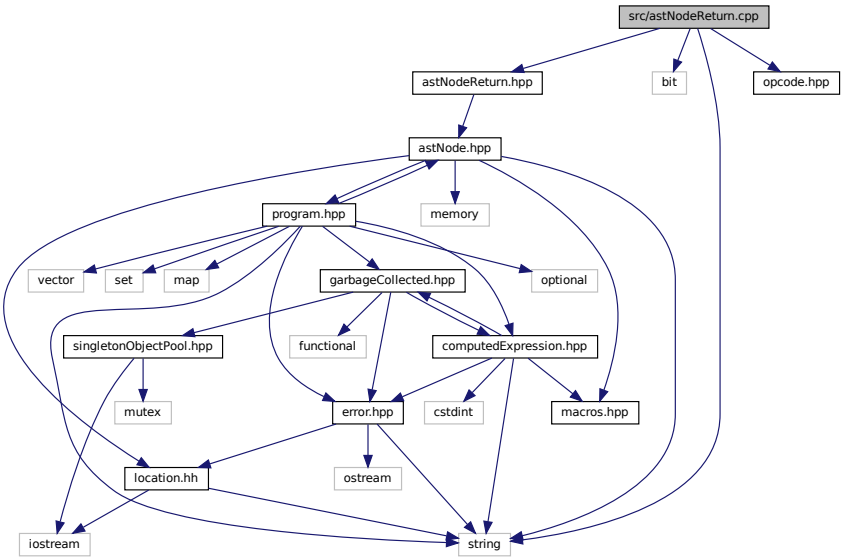
Define the Tang::AstNodePrint class.

6.53 src/astNodeReturn.cpp File Reference

Define the Tang::AstNodeReturn class.

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

Include dependency graph for astNodeReturn.cpp:



6.53.1 Detailed Description

Define the Tang::AstNodeReturn class.

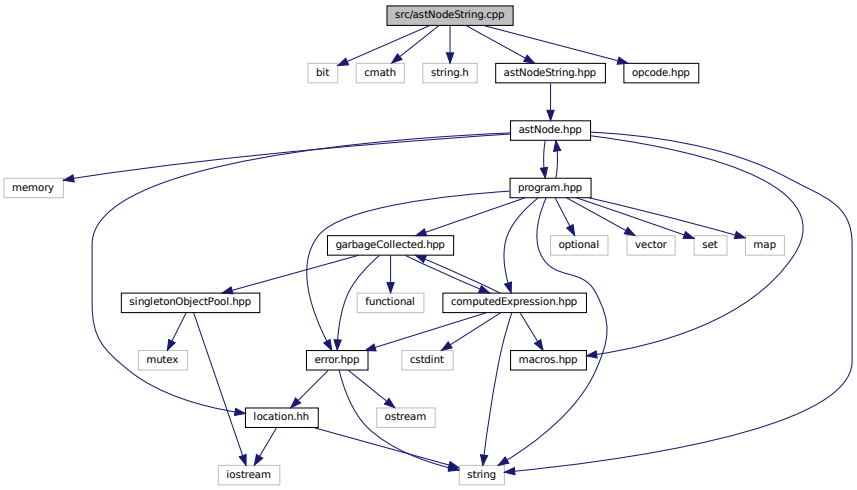
6.54 src/astNodeString.cpp File Reference

Define the Tang::AstNodeString class.

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

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

Include dependency graph for astNodeString.cpp:



6.54.1 Detailed Description

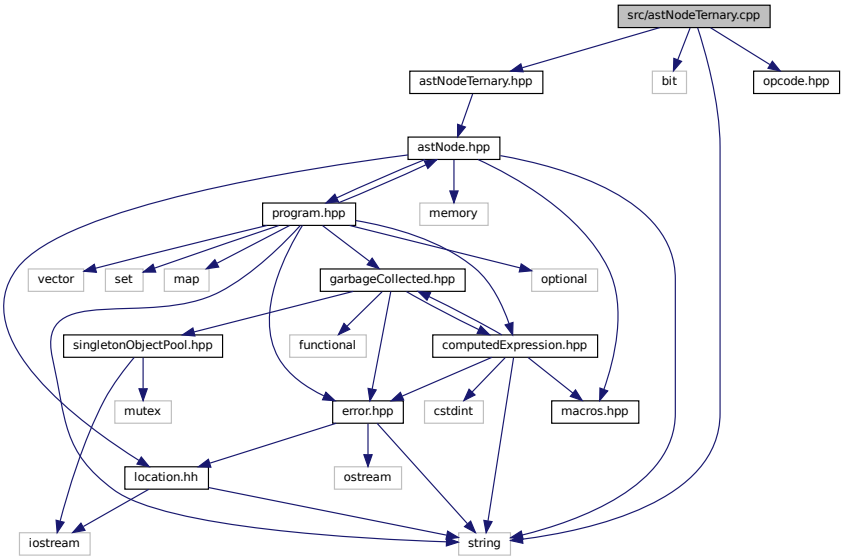
Define the Tang::AstNodeString class.

6.55 src/astNodeTernary.cpp File Reference

Define the Tang::AstNodeTernary class.

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

Include dependency graph for astNodeTernary.cpp:



6.55.1 Detailed Description

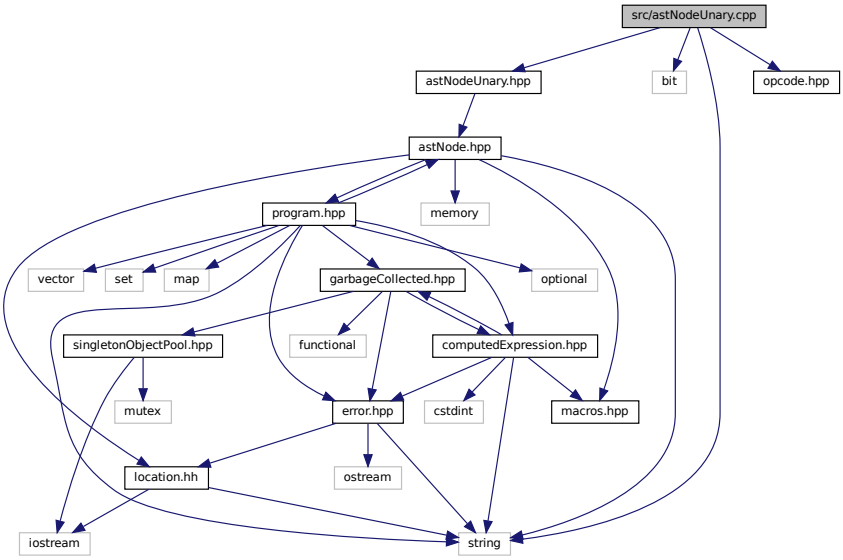
Define the Tang::AstNodeTernary class.

6.56 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:



6.56.1 Detailed Description

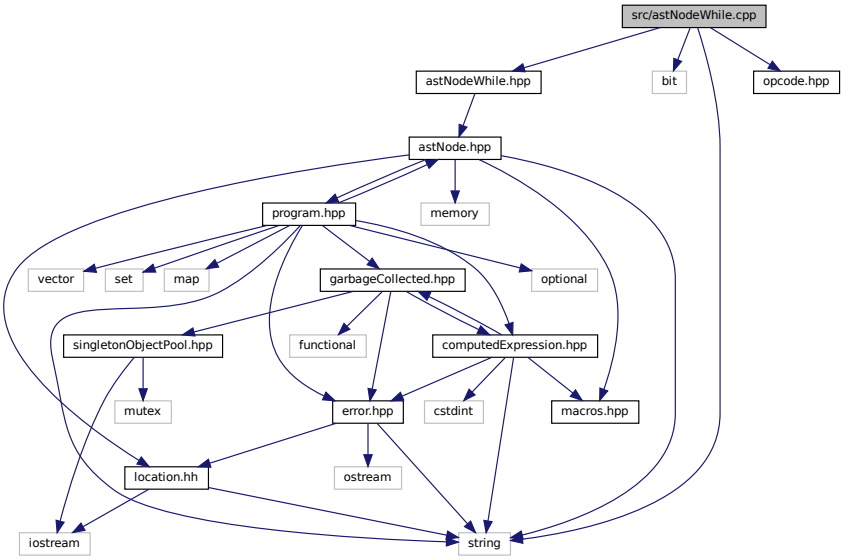
Define the Tang::AstNodeUnary class.

6.57 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:



6.57.1 Detailed Description

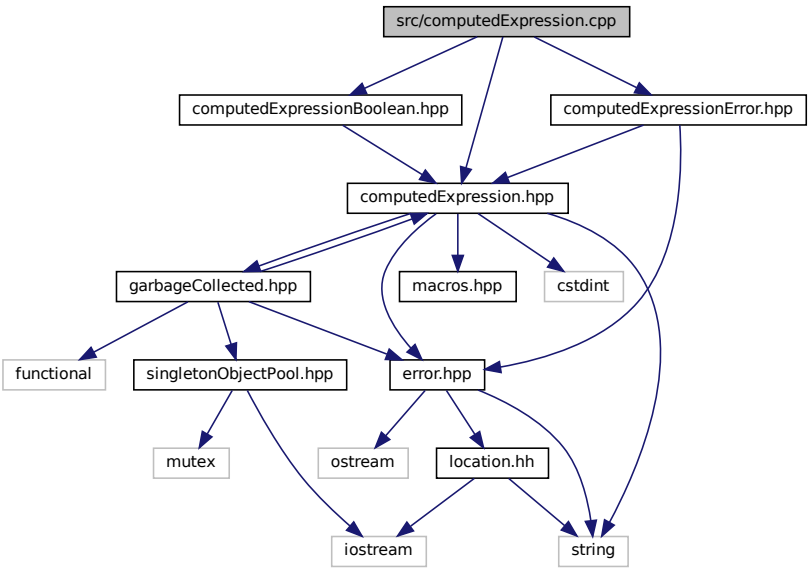
Define the Tang::AstNodeWhile class.

6.58 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:



6.58.1 Detailed Description

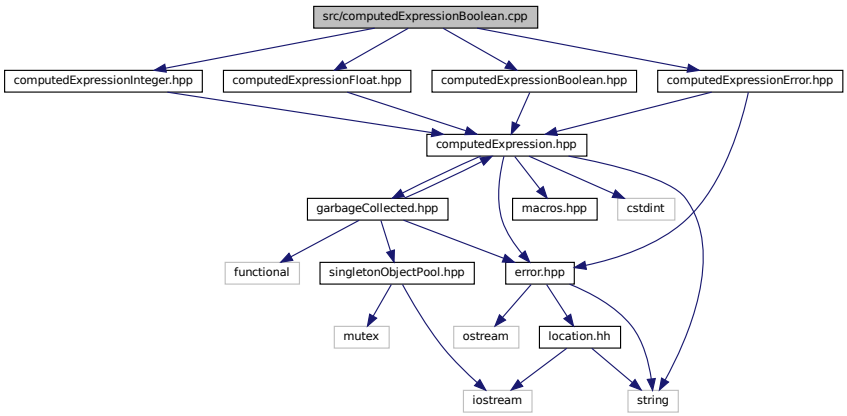
Define the Tang::ComputedExpression class.

6.59 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:



6.59.1 Detailed Description

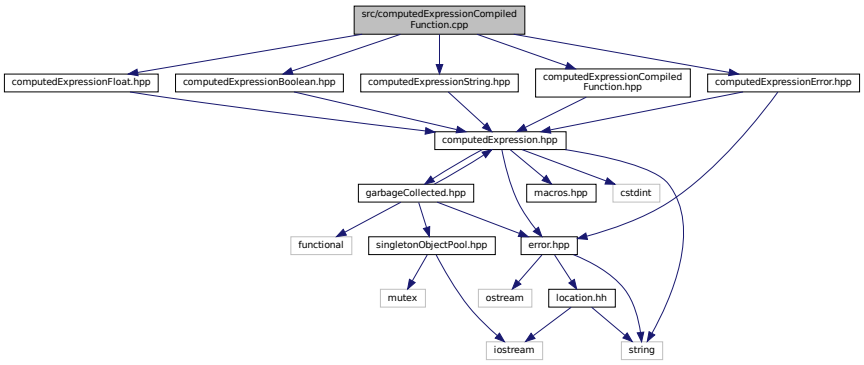
Define the Tang::ComputedExpressionBoolean class.

6.60 src/computedExpressionCompiledFunction.cpp File Reference

Define the Tang::ComputedExpressionCompiledFunction class.

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

Include dependency graph for computedExpressionCompiledFunction.cpp:



6.60.1 Detailed Description

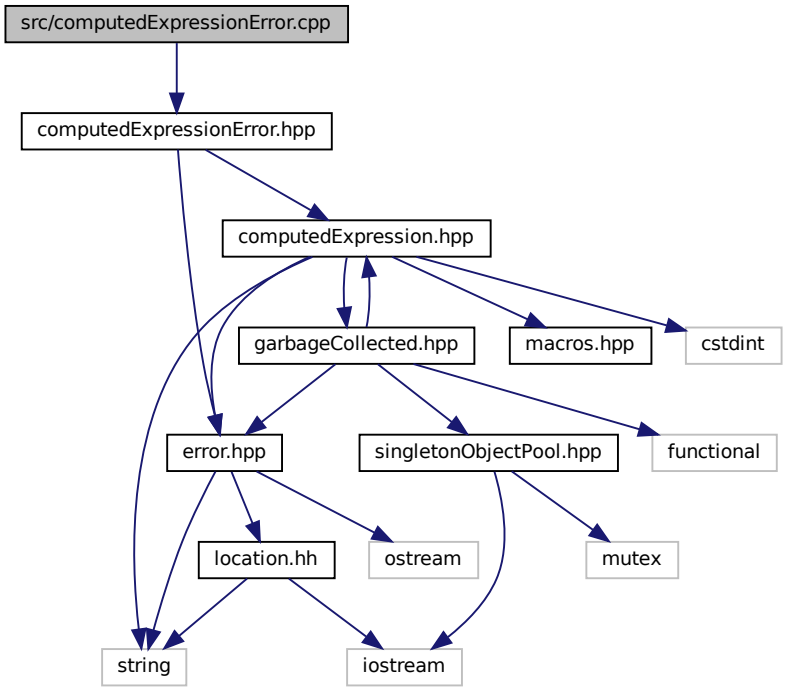
Define the Tang::ComputedExpressionCompiledFunction class.

6.61 src/computedExpressionError.cpp File Reference

Define the Tang::ComputedExpressionError class.

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

Include dependency graph for computedExpressionError.cpp:



6.61.1 Detailed Description

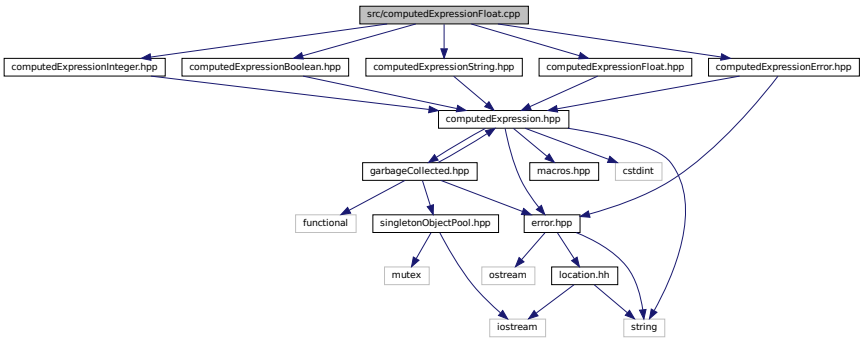
Define the Tang::ComputedExpressionError class.

6.62 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:



6.62.1 Detailed Description

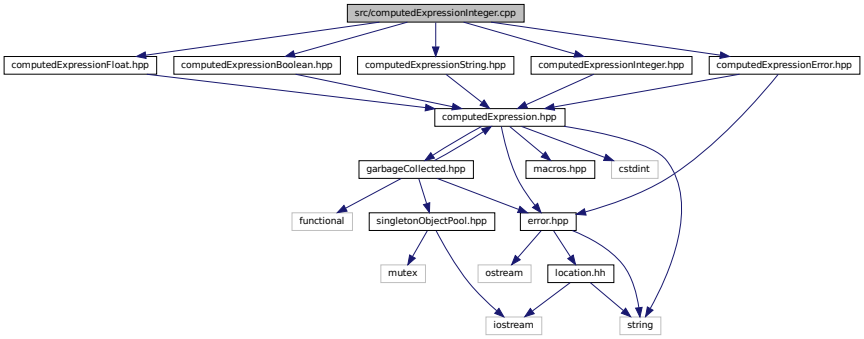
Define the Tang::ComputedExpressionFloat class.

6.63 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:



6.63.1 Detailed Description

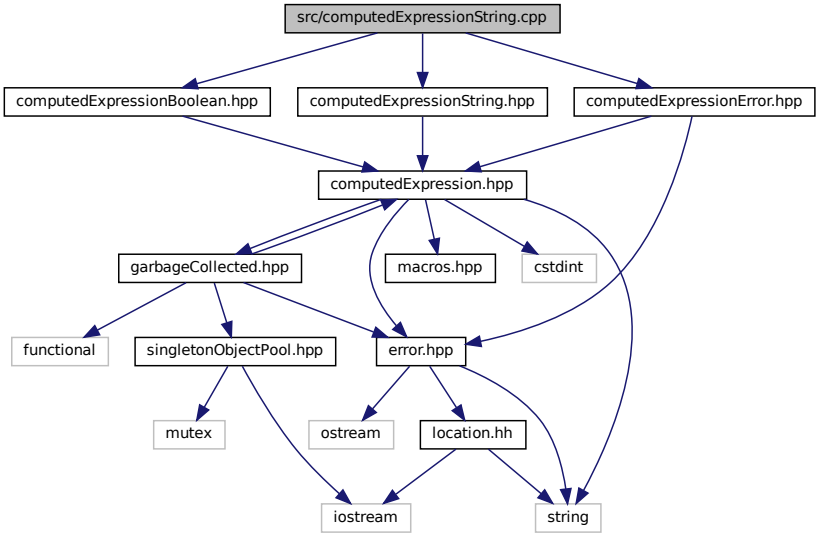
Define the Tang::ComputedExpressionInteger class.

6.64 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:



6.64.1 Detailed Description

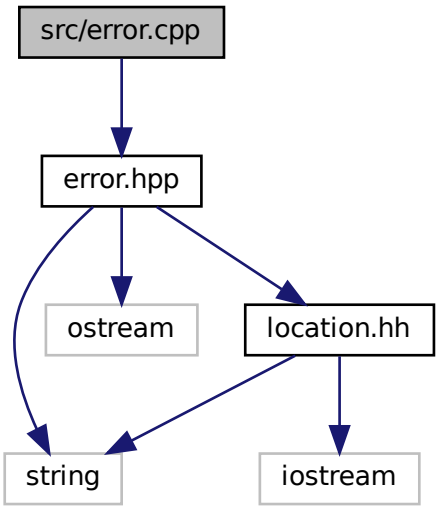
Define the Tang::ComputedExpressionString class.

6.65 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)

6.65.1 Detailed Description

Define the Tang::Error class.

6.65.2 Function Documentation

6.65.2.1 operator<<()

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

Parameters

out	The output stream.
error	The Error object.

Returns

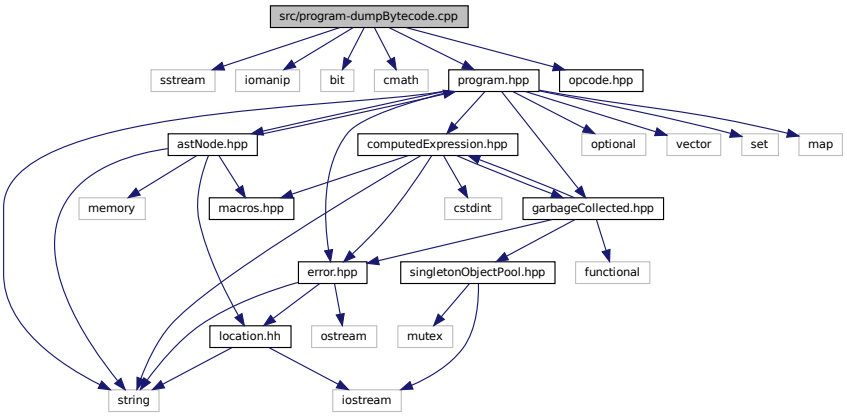
The output stream.

6.66 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.

6.66.1 Detailed Description

Define the Tang::Program::dumpBytecode method.

6.66.2 Macro Definition Documentation

6.66.2.1 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

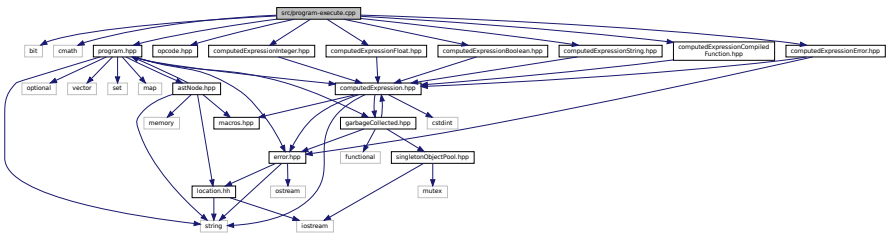
x	The number of additional vector entries that should exist.

6.67 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 "computedExpressionCompiledFunction.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.

6.67.1 Detailed Description

Define the Tang::Program::execute method.

6.67.2 Macro Definition Documentation

6.67.2.1 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.

6.67.2.2 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

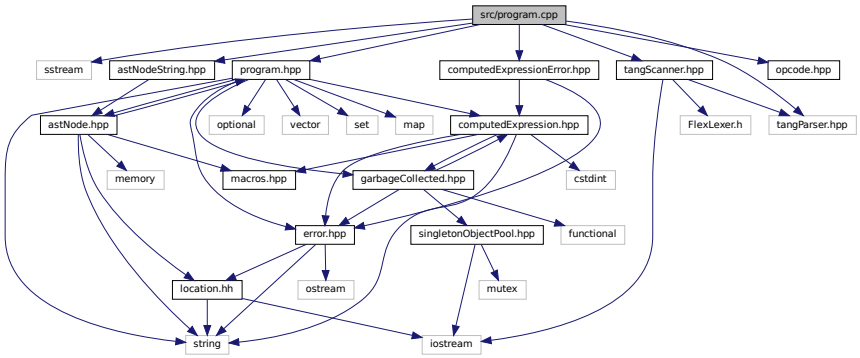
x	The number of entries that should exist in the stack.

6.68 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:



6.68.1 Detailed Description

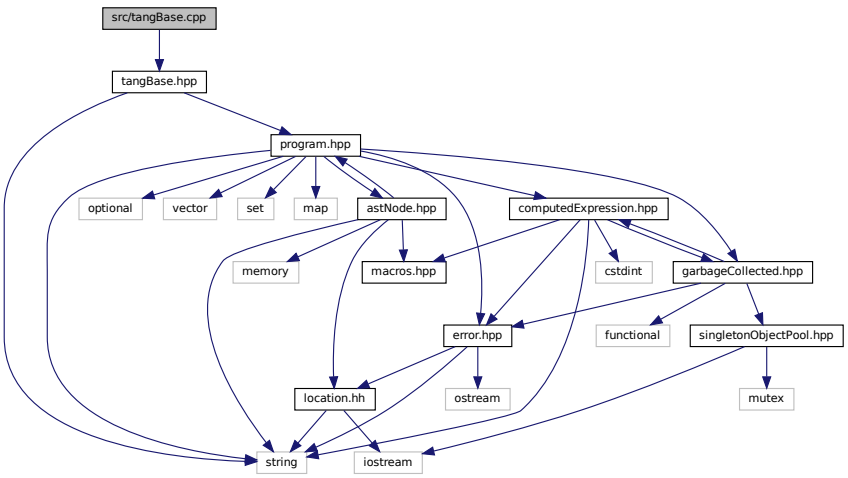
Define the Tang::Program class.

6.69 src/tangBase.cpp File Reference

Define the Tang::TangBase class.

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

Include dependency graph for tangBase.cpp:



6.69.1 Detailed Description

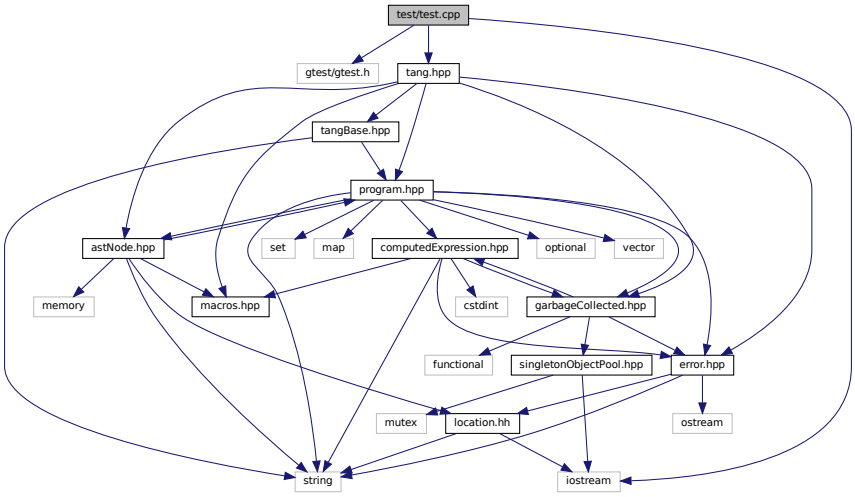
Define the Tang::TangBase class.

6.70 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** (Expression, Ternary)
- **TEST** (CodeBlock, Statements)
- **TEST** (Assign, Identifier)
- **TEST** (ControlFlow, IfElse)
- **TEST** (ControlFlow, While)
- **TEST** (ControlFlow, DoWhile)
- **TEST** (ControlFlow, For)
- **TEST** (Print, Default)
- **TEST** (Function, Compiled)
- **TEST** (Function, Recursion)
- **TEST** (Function, FunctionCall)
- **TEST** (Function, Return)
- int **main** (int argc, char ∗∗argv)

6.70.1 Detailed Description

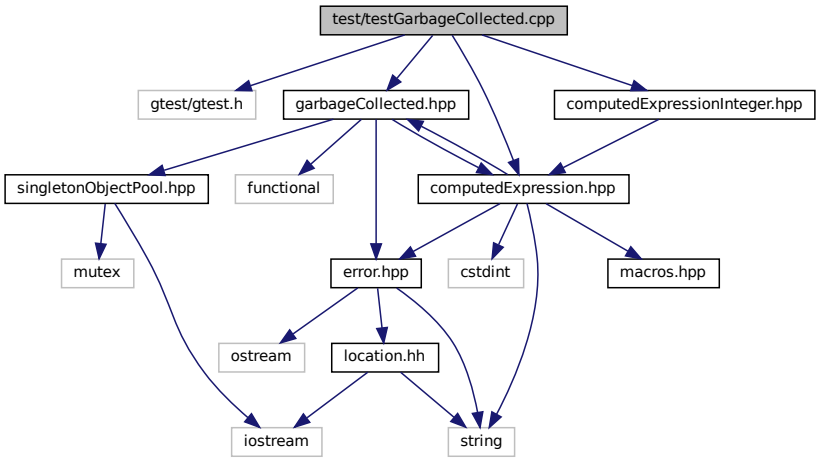
Test the general language behaviors.

6.71 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)

6.71.1 Detailed Description

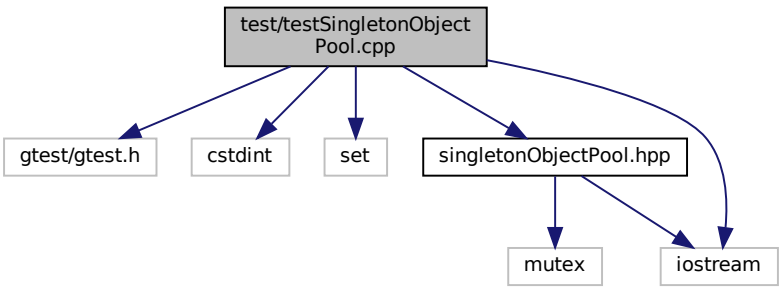
Test the generic behavior of the Tang::GarbageCollected class.

6.72 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)

6.72.1 Detailed Description

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

Index

__add
Tang::ComputedExpression, 76
Tang::ComputedExpressionBoolean, 86
Tang::ComputedExpressionCompiledFunction, 95
Tang::ComputedExpressionError, 106
Tang::ComputedExpressionFloat, 115
Tang::ComputedExpressionInteger, 125
Tang::ComputedExpressionString, 135
__boolean
Tang::ComputedExpression, 77
Tang::ComputedExpressionBoolean, 86
Tang::ComputedExpressionCompiledFunction, 96
Tang::ComputedExpressionError, 106
Tang::ComputedExpressionFloat, 116
Tang::ComputedExpressionInteger, 125
Tang::ComputedExpressionString, 135
__divide
Tang::ComputedExpression, 77
Tang::ComputedExpressionBoolean, 86
Tang::ComputedExpressionCompiledFunction, 96
Tang::ComputedExpressionError, 106
Tang::ComputedExpressionFloat, 116
Tang::ComputedExpressionInteger, 125
Tang::ComputedExpressionString, 135
__equal
Tang::ComputedExpression, 78
Tang::ComputedExpressionBoolean, 87
Tang::ComputedExpressionCompiledFunction, 96
Tang::ComputedExpressionError, 107
Tang::ComputedExpressionFloat, 116
Tang::ComputedExpressionInteger, 126
Tang::ComputedExpressionString, 136
__float
Tang::ComputedExpression, 78
Tang::ComputedExpressionBoolean, 87
Tang::ComputedExpressionCompiledFunction, 97
Tang::ComputedExpressionError, 107
Tang::ComputedExpressionFloat, 117
Tang::ComputedExpressionInteger, 126
Tang::ComputedExpressionString, 136
__integer
Tang::ComputedExpression, 78
Tang::ComputedExpressionBoolean, 87
Tang::ComputedExpressionCompiledFunction, 97
Tang::ComputedExpressionError, 107
Tang::ComputedExpressionFloat, 117
Tang::ComputedExpressionInteger, 126
Tang::ComputedExpressionString, 136
__lessThan
Tang::ComputedExpression, 78
Tang::ComputedExpressionBoolean, 88
Tang::ComputedExpressionCompiledFunction, 97
Tang::ComputedExpressionError, 108
Tang::ComputedExpressionFloat, 117
Tang::ComputedExpressionInteger, 127
Tang::ComputedExpressionString, 136
__modulo
Tang::ComputedExpression, 79
Tang::ComputedExpressionBoolean, 88
Tang::ComputedExpressionCompiledFunction, 98
Tang::ComputedExpressionError, 108
Tang::ComputedExpressionFloat, 118
Tang::ComputedExpressionInteger, 127
Tang::ComputedExpressionString, 137
__multiply
Tang::ComputedExpression, 79
Tang::ComputedExpressionBoolean, 89
Tang::ComputedExpressionCompiledFunction, 98
Tang::ComputedExpressionError, 108
Tang::ComputedExpressionFloat, 118
Tang::ComputedExpressionInteger, 127
Tang::ComputedExpressionString, 137
__negative
Tang::ComputedExpression, 79
Tang::ComputedExpressionBoolean, 89
Tang::ComputedExpressionCompiledFunction, 98
Tang::ComputedExpressionError, 109
Tang::ComputedExpressionFloat, 118
Tang::ComputedExpressionInteger, 128
Tang::ComputedExpressionString, 138
__not
Tang::ComputedExpression, 80
Tang::ComputedExpressionBoolean, 89
Tang::ComputedExpressionCompiledFunction, 99
Tang::ComputedExpressionError, 109
Tang::ComputedExpressionFloat, 119
Tang::ComputedExpressionInteger, 128
Tang::ComputedExpressionString, 138
__string
Tang::ComputedExpression, 80
Tang::ComputedExpressionBoolean, 89
Tang::ComputedExpressionCompiledFunction, 99
Tang::ComputedExpressionError, 109
Tang::ComputedExpressionFloat, 119
Tang::ComputedExpressionInteger, 128
Tang::ComputedExpressionString, 138
__subtract
Tang::ComputedExpression, 80

Tang::ComputedExpressionBoolean, 90
Tang::ComputedExpressionCompiledFunction, 99
Tang::ComputedExpressionError, 110
Tang::ComputedExpressionFloat, 119
Tang::ComputedExpressionInteger, 129
Tang::ComputedExpressionString, 138
~GarbageCollected
Tang::GarbageCollected, 147

ADD
opcode.hpp, 209
Add
Tang::AstNodeBinary, 20
addBytecode
Tang::Program, 165
addIdentifier
Tang::Program, 167
addString
Tang::Program, 167
And
Tang::AstNodeBinary, 20
AstNode
Tang::AstNode, 13
AstNodeAssign
Tang::AstNodeAssign, 16
AstNodeBinary
Tang::AstNodeBinary, 20
AstNodeBlock
Tang::AstNodeBlock, 23
AstNodeBoolean
Tang::AstNodeBoolean, 26
AstNodeCast
Tang::AstNodeCast, 29
AstNodeDoWhile
Tang::AstNodeDoWhile, 32
AstNodeFloat
Tang::AstNodeFloat, 35
AstNodeFor
Tang::AstNodeFor, 38
AstNodeFunctionCall
Tang::AstNodeFunctionCall, 41
AstNodeFunctionDeclaration
Tang::AstNodeFunctionDeclaration, 44
AstNodeIdentifier
Tang::AstNodeIdentifier, 47
AstNodeIfElse
Tang::AstNodeIfElse, 51
AstNodeInteger
Tang::AstNodeInteger, 54
AstNodePrint
Tang::AstNodePrint, 57
AstNodeReturn
Tang::AstNodeReturn, 60
AstNodeString
Tang::AstNodeString, 63
AstNodeTernary
Tang::AstNodeTernary, 67
AstNodeUnary
Tang::AstNodeUnary, 70
AstNodeWhile
Tang::AstNodeWhile, 73

BOOLEAN
opcode.hpp, 209
Boolean
Tang::AstNodeCast, 29
build/generated/location.hh, 177

CALLFUNC
opcode.hpp, 209
CASTBOOLEAN
opcode.hpp, 209
CASTFLOAT
opcode.hpp, 209
CASTINTEGER
opcode.hpp, 209
CodeType
Tang::Program, 165
compile
Tang::AstNode, 13
Tang::AstNodeAssign, 17
Tang::AstNodeBinary, 20
Tang::AstNodeBlock, 23
Tang::AstNodeBoolean, 26
Tang::AstNodeCast, 29
Tang::AstNodeDoWhile, 32
Tang::AstNodeFloat, 35
Tang::AstNodeFor, 38
Tang::AstNodeFunctionCall, 41
Tang::AstNodeFunctionDeclaration, 44
Tang::AstNodeIdentifier, 48
Tang::AstNodeIfElse, 51
Tang::AstNodeInteger, 54
Tang::AstNodePrint, 58
Tang::AstNodeReturn, 60
Tang::AstNodeString, 63
Tang::AstNodeTernary, 67
Tang::AstNodeUnary, 71
Tang::AstNodeWhile, 73
compileLiteral
Tang::AstNodeString, 64
compilePreprocess
Tang::AstNode, 14
Tang::AstNodeAssign, 17
Tang::AstNodeBinary, 21
Tang::AstNodeBlock, 24
Tang::AstNodeBoolean, 26
Tang::AstNodeCast, 30
Tang::AstNodeDoWhile, 33
Tang::AstNodeFloat, 36
Tang::AstNodeFor, 39
Tang::AstNodeFunctionCall, 42
Tang::AstNodeFunctionDeclaration, 45
Tang::AstNodeIdentifier, 48
Tang::AstNodeIfElse, 52
Tang::AstNodeInteger, 55
Tang::AstNodePrint, 58
Tang::AstNodeReturn, 61

Tang::AstNodeString, 64
Tang::AstNodeTernary, 68
Tang::AstNodeUnary, 71
Tang::AstNodeWhile, 74
compileScript
Tang::TangBase, 173
ComputedExpressionBoolean
Tang::ComputedExpressionBoolean, 86
ComputedExpressionCompiledFunction
Tang::ComputedExpressionCompiledFunction, 95
ComputedExpressionError
Tang::ComputedExpressionError, 105
ComputedExpressionFloat
Tang::ComputedExpressionFloat, 115
ComputedExpressionInteger
Tang::ComputedExpressionInteger, 124
ComputedExpressionString
Tang::ComputedExpressionString, 134

Default
Tang::AstNodePrint, 57
DIVIDE
opcode.hpp, 209
Divide
Tang::AstNodeBinary, 20
dump
Tang::AstNode, 14
Tang::AstNodeAssign, 17
Tang::AstNodeBinary, 21
Tang::AstNodeBlock, 24
Tang::AstNodeBoolean, 27
Tang::AstNodeCast, 30
Tang::AstNodeDoWhile, 33
Tang::AstNodeFloat, 36
Tang::AstNodeFor, 39
Tang::AstNodeFunctionCall, 42
Tang::AstNodeFunctionDeclaration, 45
Tang::AstNodeIdentifier, 49
Tang::AstNodeIfElse, 52
Tang::AstNodeInteger, 55
Tang::AstNodePrint, 58
Tang::AstNodeReturn, 61
Tang::AstNodeString, 65
Tang::AstNodeTernary, 68
Tang::AstNodeUnary, 71
Tang::AstNodeWhile, 74
Tang::ComputedExpression, 81
Tang::ComputedExpressionBoolean, 90
Tang::ComputedExpressionCompiledFunction, 100
Tang::ComputedExpressionError, 110
Tang::ComputedExpressionFloat, 120
Tang::ComputedExpressionInteger, 129
Tang::ComputedExpressionString, 139
dumpBytecode
Tang::Program, 167
DUMPPROGRAMCHECK
program-dumpBytecode.cpp, 237

EQ
opcode.hpp, 209
Equal
Tang::AstNodeBinary, 20
Error
Tang::Error, 143
error.cpp
operator<<, 236
execute
Tang::Program, 167
EXECUTEPROGRAMCHECK
program-execute.cpp, 239

FLOAT
opcode.hpp, 209
Float
Tang::AstNodeCast, 29
FUNCTION
opcode.hpp, 209
functionsDeclared
Tang::Program, 170

GarbageCollected
Tang::GarbageCollected, 146, 147
get
Tang::SingletonObjectPool< T >, 171
get_next_token
Tang::TangScanner, 175
getAst
Tang::Program, 168
getBytecode
Tang::Program, 168
getCode
Tang::Program, 168
getIdentifiers
Tang::Program, 169
getInstance
Tang::SingletonObjectPool< T >, 172
getResult
Tang::Program, 169
getStrings
Tang::Program, 169
GreaterThan
Tang::AstNodeBinary, 20
GreaterThanEqual
Tang::AstNodeBinary, 20
GT
opcode.hpp, 209
GTE
opcode.hpp, 209

include/astNode.hpp, 179
include/astNodeAssign.hpp, 180
include/astNodeBinary.hpp, 181
include/astNodeBlock.hpp, 182
include/astNodeBoolean.hpp, 183
include/astNodeCast.hpp, 184
include/astNodeDoWhile.hpp, 185
include/astNodeFloat.hpp, 186
include/astNodeFor.hpp, 187

include/astNodeFunctionCall.hpp, 188
include/astNodeFunctionDeclaration.hpp, 189
include/astNodeIdentifier.hpp, 190
include/astNodeIfElse.hpp, 191
include/astNodeInteger.hpp, 192
include/astNodePrint.hpp, 193
include/astNodeReturn.hpp, 194
include/astNodeString.hpp, 195
include/astNodeTernary.hpp, 196
include/astNodeUnary.hpp, 197
include/astNodeWhile.hpp, 198
include/computedExpression.hpp, 199
include/computedExpressionBoolean.hpp, 200
include/computedExpressionCompiledFunction.hpp, 201
include/computedExpressionError.hpp, 202
include/computedExpressionFloat.hpp, 203
include/computedExpressionInteger.hpp, 204
include/computedExpressionString.hpp, 205
include/error.hpp, 206
include/garbageCollected.hpp, 207
include/macros.hpp, 207
include/opcode.hpp, 208
include/program.hpp, 209
include/singletonObjectPool.hpp, 210
include/tang.hpp, 211
include/tangBase.hpp, 212
include/tangScanner.hpp, 214
INTEGER
    opcode.hpp, 209
Integer
    Tang::AstNodeCast, 29
is_equal
    Tang::ComputedExpression, 81–83
    Tang::ComputedExpressionBoolean, 90–92
    Tang::ComputedExpressionCompiledFunction, 100, 102, 103
    Tang::ComputedExpressionError, 110–112
    Tang::ComputedExpressionFloat, 120–122
    Tang::ComputedExpressionInteger, 130, 131
    Tang::ComputedExpressionString, 139–141

JMP
    opcode.hpp, 209
JMPF
    opcode.hpp, 209
JMPF_POP
    opcode.hpp, 209
JMPT
    opcode.hpp, 209
JMPT_POP
    opcode.hpp, 209

LessThan
    Tang::AstNodeBinary, 20
LessThanEqual
    Tang::AstNodeBinary, 20
location.hh
    operator<<, 178, 179
LT
    opcode.hpp, 209
LTE
    opcode.hpp, 209

make
    Tang::GarbageCollected, 147
makeCopy
    Tang::ComputedExpression, 83
    Tang::ComputedExpressionBoolean, 93
    Tang::ComputedExpressionCompiledFunction, 103
    Tang::ComputedExpressionError, 113
    Tang::ComputedExpressionFloat, 122
    Tang::ComputedExpressionInteger, 132
    Tang::ComputedExpressionString, 141
MODULO
    opcode.hpp, 209
Modulo
    Tang::AstNodeBinary, 20
MULTIPLY
    opcode.hpp, 209
Multiply
    Tang::AstNodeBinary, 20

NEGATIVE
    opcode.hpp, 209
Negative
    Tang::AstNodeUnary, 70
NEQ
    opcode.hpp, 209
NOT
    opcode.hpp, 209
Not
    Tang::AstNodeUnary, 70
NotEqual
    Tang::AstNodeBinary, 20
NULLVAL
    opcode.hpp, 209

Opcode
    opcode.hpp, 208
opcode.hpp
    ADD, 209
    BOOLEAN, 209
    CALLFUNC, 209
    CASTBOOLEAN, 209
    CASTFLOAT, 209
    CASTINTEGER, 209
    DIVIDE, 209
    EQ, 209
    FLOAT, 209
    FUNCTION, 209
    GT, 209
    GTE, 209
    INTEGER, 209
    JMP, 209
    JMPF, 209
    JMPF_POP, 209
    JMPT, 209

JMPT_POP, 209
LT, 209
LTE, 209
MODULO, 209
MULTIPLY, 209
NEGATIVE, 209
NEQ, 209
NOT, 209
NULLVAL, 209
Opcode, 208
PEEK, 209
POKE, 209
POP, 209
PRINT, 209
RETURN, 209
STRING, 209
SUBTRACT, 209
Operation
Tang::AstNodeBinary, 19
Operator
Tang::AstNodeUnary, 70
operator!
Tang::GarbageCollected, 148
operator!=
Tang::GarbageCollected, 148
operator<
Tang::GarbageCollected, 153
operator<<
error.cpp, 236
location.hh, 178, 179
Tang::Error, 143
Tang::GarbageCollected, 160
operator<=
Tang::GarbageCollected, 153
operator>
Tang::GarbageCollected, 159
operator>=
Tang::GarbageCollected, 159
operator*
Tang::GarbageCollected, 149, 150
operator+
Tang::GarbageCollected, 150
operator-
Tang::GarbageCollected, 151
operator->
Tang::GarbageCollected, 152
operator/
Tang::GarbageCollected, 152
operator=
Tang::GarbageCollected, 154
operator==
Tang::GarbageCollected, 155–157
operator%
Tang::GarbageCollected, 149
Or
Tang::AstNodeBinary, 20

PEEK
opcode.hpp, 209
POKE
opcode.hpp, 209
POP
opcode.hpp, 209
PRINT
opcode.hpp, 209
Program
Tang::Program, 165
program-dumpBytecode.cpp
DUMPPROGRAMCHECK, 237
program-execute.cpp
EXECUTEPROGRAMCHECK, 239
STACKCHECK, 239
pushEnvironment
Tang::Program, 169

recycle
Tang::SingletonObjectPool< T >, 172
RETURN
opcode.hpp, 209

Script
Tang::Program, 165
setFunctionStackDeclaration
Tang::Program, 170
setJumpTarget
Tang::Program, 170
src/astNode.cpp, 215
src/astNodeAssign.cpp, 215
src/astNodeBinary.cpp, 216
src/astNodeBlock.cpp, 217
src/astNodeBoolean.cpp, 217
src/astNodeCast.cpp, 218
src/astNodeDoWhile.cpp, 219
src/astNodeFloat.cpp, 219
src/astNodeFor.cpp, 220
src/astNodeFunctionCall.cpp, 221
src/astNodeFunctionDeclaration.cpp, 222
src/astNodeIdentifier.cpp, 223
src/astNodeIfElse.cpp, 224
src/astNodeInteger.cpp, 224
src/astNodePrint.cpp, 225
src/astNodeReturn.cpp, 226
src/astNodeString.cpp, 226
src/astNodeTernary.cpp, 227
src/astNodeUnary.cpp, 228
src/astNodeWhile.cpp, 229
src/computedExpression.cpp, 230
src/computedExpressionBoolean.cpp, 231
src/computedExpressionCompiledFunction.cpp, 232
src/computedExpressionError.cpp, 232
src/computedExpressionFloat.cpp, 233
src/computedExpressionInteger.cpp, 234
src/computedExpressionString.cpp, 235
src/error.cpp, 235
src/program-dumpBytecode.cpp, 237
src/program-execute.cpp, 238
src/program.cpp, 239
src/tangBase.cpp, 240

STACKCHECK
program-execute.cpp, 239
STRING
opcode.hpp, 209
SUBTRACT
opcode.hpp, 209
Subtract
Tang::AstNodeBinary, 20

Tang::AstNode, 11
AstNode, 13
compile, 13
compilePreprocess, 14
dump, 14
Tang::AstNodeAssign, 15
AstNodeAssign, 16
compile, 17
compilePreprocess, 17
dump, 17
Tang::AstNodeBinary, 18
Add, 20
And, 20
AstNodeBinary, 20
compile, 20
compilePreprocess, 21
Divide, 20
dump, 21
Equal, 20
GreaterThan, 20
GreaterThanEqual, 20
LessThan, 20
LessThanEqual, 20
Modulo, 20
Multiply, 20
NotEqual, 20
Operation, 19
Or, 20
Subtract, 20
Tang::AstNodeBlock, 22
AstNodeBlock, 23
compile, 23
compilePreprocess, 24
dump, 24
Tang::AstNodeBoolean, 25
AstNodeBoolean, 26
compile, 26
compilePreprocess, 26
dump, 27
Tang::AstNodeCast, 27
AstNodeCast, 29
Boolean, 29
compile, 29
compilePreprocess, 30
dump, 30
Float, 29
Integer, 29
Type, 29
Tang::AstNodeDoWhile, 31
AstNodeDoWhile, 32
compile, 32
compilePreprocess, 33
dump, 33
Tang::AstNodeFloat, 34
AstNodeFloat, 35
compile, 35
compilePreprocess, 36
dump, 36
Tang::AstNodeFor, 37
AstNodeFor, 38
compile, 38
compilePreprocess, 39
dump, 39
Tang::AstNodeFunctionCall, 40
AstNodeFunctionCall, 41
compile, 41
compilePreprocess, 42
dump, 42
Tang::AstNodeFunctionDeclaration, 43
AstNodeFunctionDeclaration, 44
compile, 44
compilePreprocess, 45
dump, 45
Tang::AstNodeIdentifier, 46
AstNodeIdentifier, 47
compile, 48
compilePreprocess, 48
dump, 49
Tang::AstNodeIfElse, 49
AstNodeIfElse, 51
compile, 51
compilePreprocess, 52
dump, 52
Tang::AstNodeInteger, 53
AstNodeInteger, 54
compile, 54
compilePreprocess, 55
dump, 55
Tang::AstNodePrint, 56
AstNodePrint, 57
compile, 58
compilePreprocess, 58
Default, 57
dump, 58
Type, 57
Tang::AstNodeReturn, 59
AstNodeReturn, 60
compile, 60
compilePreprocess, 61
dump, 61
Tang::AstNodeString, 62
AstNodeString, 63
compile, 63
compileLiteral, 64
compilePreprocess, 64
dump, 65
Tang::AstNodeTernary, 66
AstNodeTernary, 67

compile, 67
compilePreprocess, 68
dump, 68
Tang::AstNodeUnary, 69
AstNodeUnary, 70
compile, 71
compilePreprocess, 71
dump, 71
Negative, 70
Not, 70
Operator, 70
Tang::AstNodeWhile, 72
AstNodeWhile, 73
compile, 73
compilePreprocess, 74
dump, 74
Tang::ComputedExpression, 75
__add, 76
__boolean, 77
__divide, 77
__equal, 78
__float, 78
__integer, 78
__lessThan, 78
__modulo, 79
__multiply, 79
__negative, 79
__not, 80
__string, 80
__subtract, 80
dump, 81
is_equal, 81–83
makeCopy, 83
Tang::ComputedExpressionBoolean, 84
__add, 86
__boolean, 86
__divide, 86
__equal, 87
__float, 87
__integer, 87
__lessThan, 88
__modulo, 88
__multiply, 89
__negative, 89
__not, 89
__string, 89
__subtract, 90
ComputedExpressionBoolean, 86
dump, 90
is_equal, 90–92
makeCopy, 93
Tang::ComputedExpressionCompiledFunction, 93
__add, 95
__boolean, 96
__divide, 96
__equal, 96
__float, 97
__integer, 97
__lessThan, 97
__modulo, 98
__multiply, 98
__negative, 98
__not, 99
__string, 99
__subtract, 99
ComputedExpressionCompiledFunction, 95
dump, 100
is_equal, 100, 102, 103
makeCopy, 103
Tang::ComputedExpressionError, 104
__add, 106
__boolean, 106
__divide, 106
__equal, 107
__float, 107
__integer, 107
__lessThan, 108
__modulo, 108
__multiply, 108
__negative, 109
__not, 109
__string, 109
__subtract, 110
ComputedExpressionError, 105
dump, 110
is_equal, 110–112
makeCopy, 113
Tang::ComputedExpressionFloat, 113
__add, 115
__boolean, 116
__divide, 116
__equal, 116
__float, 117
__integer, 117
__lessThan, 117
__modulo, 118
__multiply, 118
__negative, 118
__not, 119
__string, 119
__subtract, 119
ComputedExpressionFloat, 115
dump, 120
is_equal, 120–122
makeCopy, 122
Tang::ComputedExpressionInteger, 123
__add, 125
__boolean, 125
__divide, 125
__equal, 126
__float, 126
__integer, 126
__lessThan, 127
__modulo, 127
__multiply, 127
__negative, 128

__not, 128
__string, 128
__subtract, 129
ComputedExpressionInteger, 124
dump, 129
is_equal, 130, 131
makeCopy, 132
Tang::ComputedExpressionString, 132
__add, 135
__boolean, 135
__divide, 135
__equal, 136
__float, 136
__integer, 136
__lessThan, 136
__modulo, 137
__multiply, 137
__negative, 138
__not, 138
__string, 138
__subtract, 138
ComputedExpressionString, 134
dump, 139
is_equal, 139–141
makeCopy, 141
Tang::Error, 142
Error, 143
operator<<, 143
Tang::GarbageCollected, 144
∼GarbageCollected, 147
GarbageCollected, 146, 147
make, 147
operator!, 148
operator!=, 148
operator<, 153
operator<<, 160
operator<=, 153
operator>, 159
operator>=, 159
operator∗, 149, 150
operator+, 150
operator-, 151
operator->, 152
operator/, 152
operator=, 154
operator==, 155–157
operator%, 149
Tang::location, 160
Tang::position, 162
Tang::Program, 163
addBytecode, 165
addIdentifier, 167
addString, 167
CodeType, 165
dumpBytecode, 167
execute, 167
functionsDeclared, 170
getAst, 168
getBytecode, 168
getCode, 168
getIdentifiers, 169
getResult, 169
getStrings, 169
Program, 165
pushEnvironment, 169
Script, 165
setFunctionStackDeclaration, 170
setJumpTarget, 170
Template, 165
Tang::SingletonObjectPool< T >, 171
get, 171
getInstance, 172
recycle, 172
Tang::TangBase, 172
compileScript, 173
TangBase, 173
Tang::TangScanner, 174
get_next_token, 175
TangScanner, 175
TangBase
Tang::TangBase, 173
TangScanner
Tang::TangScanner, 175
Template
Tang::Program, 165
test/test.cpp, 241
test/testGarbageCollected.cpp, 242
test/testSingletonObjectPool.cpp, 243
Type
Tang::AstNodeCast, 29
Tang::AstNodePrint, 57