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 collectFunctionDeclarations() . . . 13
5.1.3.2 collectIdentifiers() . . . 14
5.1.3.3 collectStrings() . . . 14
5.1.3.4 compile() . . . 14
5.1.3.5 dump() . . . 15
5.2 Tang::AstNodeAssign Class Reference . . . 16
5.2.1 Detailed Description . . . 17
5.2.2 Constructor & Destructor Documentation . . . 17
5.2.2.1 AstNodeAssign() . . . 17
5.2.3 Member Function Documentation . . . 17
5.2.3.1 collectFunctionDeclarations() . . . 17
5.2.3.2 collectIdentifiers() . . . 19
5.2.3.3 collectStrings() . . . 19
5.2.3.4 compile() . . . 19
5.2.3.5 dump() . . . 20
5.3 Tang::AstNodeBinary Class Reference . . . 20
5.3.1 Detailed Description . . . 22
5.3.2 Member Enumeration Documentation . . . 22
5.3.2.1 Operation . . . 22
5.3.3 Constructor & Destructor Documentation . . . 22
5.3.3.1 AstNodeBinary() . . . 22
5.3.4 Member Function Documentation . . . 23
5.3.4.1 collectFunctionDeclarations() . . . 23
5.3.4.2 collectIdentifiers() . . . 23

5.3.4.3 collectStrings() . . . 24
5.3.4.4 compile() . . . 24
5.3.4.5 dump() . . . 24
5.4 Tang::AstNodeBlock Class Reference . . . 25
5.4.1 Detailed Description . . . 26
5.4.2 Constructor & Destructor Documentation . . . 26
5.4.2.1 AstNodeBlock() . . . 26
5.4.3 Member Function Documentation . . . 26
5.4.3.1 collectFunctionDeclarations() . . . 26
5.4.3.2 collectIdentifiers() . . . 27
5.4.3.3 collectStrings() . . . 27
5.4.3.4 compile() . . . 27
5.4.3.5 dump() . . . 28
5.5 Tang::AstNodeBoolean Class Reference . . . 28
5.5.1 Detailed Description . . . 29
5.5.2 Constructor & Destructor Documentation . . . 30
5.5.2.1 AstNodeBoolean() . . . 30
5.5.3 Member Function Documentation . . . 30
5.5.3.1 collectFunctionDeclarations() . . . 30
5.5.3.2 collectIdentifiers() . . . 30
5.5.3.3 collectStrings() . . . 31
5.5.3.4 compile() . . . 31
5.5.3.5 dump() . . . 32
5.6 Tang::AstNodeCast Class Reference . . . 32
5.6.1 Detailed Description . . . 33
5.6.2 Member Enumeration Documentation . . . 33
5.6.2.1 Type . . . 33
5.6.3 Constructor & Destructor Documentation . . . 34
5.6.3.1 AstNodeCast() . . . 34
5.6.4 Member Function Documentation . . . 34
5.6.4.1 collectFunctionDeclarations() . . . 34
5.6.4.2 collectIdentifiers() . . . 35
5.6.4.3 collectStrings() . . . 35
5.6.4.4 compile() . . . 35
5.6.4.5 dump() . . . 36
5.7 Tang::AstNodeDoWhile Class Reference . . . 36
5.7.1 Detailed Description . . . 37
5.7.2 Constructor & Destructor Documentation . . . 38
5.7.2.1 AstNodeDoWhile() . . . 38
5.7.3 Member Function Documentation . . . 38
5.7.3.1 collectFunctionDeclarations() . . . 38
5.7.3.2 collectIdentifiers() . . . 38

5.7.3.3 collectStrings() . . . 39
5.7.3.4 compile() . . . 39
5.7.3.5 dump() . . . 40
5.8 Tang::AstNodeFloat Class Reference . . . 40
5.8.1 Detailed Description . . . 41
5.8.2 Constructor & Destructor Documentation . . . 41
5.8.2.1 AstNodeFloat() . . . 41
5.8.3 Member Function Documentation . . . 42
5.8.3.1 collectFunctionDeclarations() . . . 42
5.8.3.2 collectIdentifiers() . . . 42
5.8.3.3 collectStrings() . . . 42
5.8.3.4 compile() . . . 43
5.8.3.5 dump() . . . 43
5.9 Tang::AstNodeFor Class Reference . . . 44
5.9.1 Detailed Description . . . 45
5.9.2 Constructor & Destructor Documentation . . . 45
5.9.2.1 AstNodeFor() . . . 45
5.9.3 Member Function Documentation . . . 45
5.9.3.1 collectFunctionDeclarations() . . . 46
5.9.3.2 collectIdentifiers() . . . 46
5.9.3.3 collectStrings() . . . 46
5.9.3.4 compile() . . . 46
5.9.3.5 dump() . . . 48
5.10 Tang::AstNodeFunctionCall Class Reference . . . 49
5.10.1 Detailed Description . . . 50
5.10.2 Constructor & Destructor Documentation . . . 50
5.10.2.1 AstNodeFunctionCall() . . . 50
5.10.3 Member Function Documentation . . . 50
5.10.3.1 collectFunctionDeclarations() . . . 50
5.10.3.2 collectIdentifiers() . . . 50
5.10.3.3 collectStrings() . . . 52
5.10.3.4 compile() . . . 52
5.10.3.5 dump() . . . 53
5.11 Tang::AstNodeFunctionDeclaration Class Reference . . . 53
5.11.1 Detailed Description . . . 54
5.11.2 Constructor & Destructor Documentation . . . 54
5.11.2.1 AstNodeFunctionDeclaration() . . . 54
5.11.3 Member Function Documentation . . . 55
5.11.3.1 collectFunctionDeclarations() . . . 55
5.11.3.2 collectIdentifiers() . . . 55
5.11.3.3 collectStrings() . . . 56
5.11.3.4 compile() . . . 56

5.11.3.5 dump() . . . 57
5.12 Tang::AstNodeIdentifier Class Reference . . . 57
5.12.1 Detailed Description . . . 58
5.12.2 Constructor & Destructor Documentation . . . 59
5.12.2.1 AstNodeIdentifier() . . . 59
5.12.3 Member Function Documentation . . . 59
5.12.3.1 collectFunctionDeclarations() . . . 59
5.12.3.2 collectIdentifiers() . . . 59
5.12.3.3 collectStrings() . . . 60
5.12.3.4 compile() . . . 60
5.12.3.5 dump() . . . 61
5.13 Tang::AstNodeIfElse Class Reference . . . 61
5.13.1 Detailed Description . . . 63
5.13.2 Constructor & Destructor Documentation . . . 63
5.13.2.1 AstNodeIfElse() [1/2] . . . 63
5.13.2.2 AstNodeIfElse() [2/2] . . . 63
5.13.3 Member Function Documentation . . . 63
5.13.3.1 collectFunctionDeclarations() . . . 64
5.13.3.2 collectIdentifiers() . . . 64
5.13.3.3 collectStrings() . . . 64
5.13.3.4 compile() . . . 64
5.13.3.5 dump() . . . 66
5.14 Tang::AstNodeInteger Class Reference . . . 67
5.14.1 Detailed Description . . . 68
5.14.2 Constructor & Destructor Documentation . . . 68
5.14.2.1 AstNodeInteger() . . . 68
5.14.3 Member Function Documentation . . . 68
5.14.3.1 collectFunctionDeclarations() . . . 68
5.14.3.2 collectIdentifiers() . . . 68
5.14.3.3 collectStrings() . . . 69
5.14.3.4 compile() . . . 69
5.14.3.5 dump() . . . 70
5.15 Tang::AstNodePrint Class Reference . . . 70
5.15.1 Detailed Description . . . 72
5.15.2 Member Enumeration Documentation . . . 72
5.15.2.1 Type . . . 72
5.15.3 Constructor & Destructor Documentation . . . 72
5.15.3.1 AstNodePrint() . . . 72
5.15.4 Member Function Documentation . . . 72
5.15.4.1 collectFunctionDeclarations() . . . 72
5.15.4.2 collectIdentifiers() . . . 73
5.15.4.3 collectStrings() . . . 73

5.15.4.4 compile() . . . 73
5.15.4.5 dump() . . . 74
5.16 Tang::AstNodeReturn Class Reference . . . 74
5.16.1 Detailed Description . . . 75
5.16.2 Constructor & Destructor Documentation . . . 76
5.16.2.1 AstNodeReturn() . . . 76
5.16.3 Member Function Documentation . . . 76
5.16.3.1 collectFunctionDeclarations() . . . 76
5.16.3.2 collectIdentifiers() . . . 76
5.16.3.3 collectStrings() . . . 77
5.16.3.4 compile() . . . 77
5.16.3.5 dump() . . . 78
5.17 Tang::AstNodeString Class Reference . . . 78
5.17.1 Detailed Description . . . 79
5.17.2 Constructor & Destructor Documentation . . . 79
5.17.2.1 AstNodeString() . . . 79
5.17.3 Member Function Documentation . . . 80
5.17.3.1 collectFunctionDeclarations() . . . 80
5.17.3.2 collectIdentifiers() . . . 80
5.17.3.3 collectStrings() . . . 80
5.17.3.4 compile() . . . 81
5.17.3.5 compileLiteral() . . . 82
5.17.3.6 dump() . . . 82
5.18 Tang::AstNodeTernary Class Reference . . . 83
5.18.1 Detailed Description . . . 84
5.18.2 Constructor & Destructor Documentation . . . 84
5.18.2.1 AstNodeTernary() . . . 84
5.18.3 Member Function Documentation . . . 84
5.18.3.1 collectFunctionDeclarations() . . . 84
5.18.3.2 collectIdentifiers() . . . 85
5.18.3.3 collectStrings() . . . 85
5.18.3.4 compile() . . . 85
5.18.3.5 dump() . . . 86
5.19 Tang::AstNodeUnary Class Reference . . . 86
5.19.1 Detailed Description . . . 88
5.19.2 Member Enumeration Documentation . . . 88
5.19.2.1 Operator . . . 88
5.19.3 Constructor & Destructor Documentation . . . 88
5.19.3.1 AstNodeUnary() . . . 88
5.19.4 Member Function Documentation . . . 88
5.19.4.1 collectFunctionDeclarations() . . . 89
5.19.4.2 collectIdentifiers() . . . 89

5.19.4.3 collectStrings() . . . 89
5.19.4.4 compile() . . . 89
5.19.4.5 dump() . . . 91
5.20 Tang::AstNodeWhile Class Reference . . . 91
5.20.1 Detailed Description . . . 92
5.20.2 Constructor & Destructor Documentation . . . 93
5.20.2.1 AstNodeWhile() . . . 93
5.20.3 Member Function Documentation . . . 93
5.20.3.1 collectFunctionDeclarations() . . . 93
5.20.3.2 collectIdentifiers() . . . 93
5.20.3.3 collectStrings() . . . 94
5.20.3.4 compile() . . . 94
5.20.3.5 dump() . . . 95
5.21 Tang::ComputedExpression Class Reference . . . 95
5.21.1 Detailed Description . . . 97
5.21.2 Member Function Documentation . . . 97
5.21.2.1 __add() . . . 97
5.21.2.2 __boolean() . . . 98
5.21.2.3 __divide() . . . 98
5.21.2.4 __equal() . . . 98
5.21.2.5 __float() . . . 99
5.21.2.6 __integer() . . . 99
5.21.2.7 __lessThan() . . . 99
5.21.2.8 __modulo() . . . 100
5.21.2.9 __multiply() . . . 100
5.21.2.10 __negative() . . . 101
5.21.2.11 __not() . . . 101
5.21.2.12 __string() . . . 101
5.21.2.13 __subtract() . . . 101
5.21.2.14 dump() . . . 102
5.21.2.15 is_equal() [1/6] . . . 102
5.21.2.16 is_equal() [2/6] . . . 103
5.21.2.17 is_equal() [3/6] . . . 103
5.21.2.18 is_equal() [4/6] . . . 103
5.21.2.19 is_equal() [5/6] . . . 104
5.21.2.20 is_equal() [6/6] . . . 104
5.21.2.21 makeCopy() . . . 104
5.22 Tang::ComputedExpressionBoolean Class Reference . . . 105
5.22.1 Detailed Description . . . 106
5.22.2 Constructor & Destructor Documentation . . . 107
5.22.2.1 ComputedExpressionBoolean() . . . 107
5.22.3 Member Function Documentation . . . 107

5.22.3.1 __add() . . . 107
5.22.3.2 __boolean() . . . 107
5.22.3.3 __divide() . . . 108
5.22.3.4 __equal() . . . 108
5.22.3.5 __float() . . . 108
5.22.3.6 __integer() . . . 109
5.22.3.7 __lessThan() . . . 109
5.22.3.8 __modulo() . . . 109
5.22.3.9 __multiply() . . . 110
5.22.3.10 __negative() . . . 110
5.22.3.11 __not() . . . 110
5.22.3.12 __string() . . . 111
5.22.3.13 __subtract() . . . 111
5.22.3.14 dump() . . . 111
5.22.3.15 is_equal() [1/6] . . . 111
5.22.3.16 is_equal() [2/6] . . . 112
5.22.3.17 is_equal() [3/6] . . . 112
5.22.3.18 is_equal() [4/6] . . . 113
5.22.3.19 is_equal() [5/6] . . . 113
5.22.3.20 is_equal() [6/6] . . . 113
5.22.3.21 makeCopy() . . . 114
5.23 Tang::ComputedExpressionCompiledFunction Class Reference . . . 114
5.23.1 Detailed Description . . . 116
5.23.2 Constructor & Destructor Documentation . . . 116
5.23.2.1 ComputedExpressionCompiledFunction() . . . 116
5.23.3 Member Function Documentation . . . 116
5.23.3.1 __add() . . . 116
5.23.3.2 __boolean() . . . 117
5.23.3.3 __divide() . . . 117
5.23.3.4 __equal() . . . 118
5.23.3.5 __float() . . . 118
5.23.3.6 __integer() . . . 118
5.23.3.7 __lessThan() . . . 118
5.23.3.8 __modulo() . . . 119
5.23.3.9 __multiply() . . . 119
5.23.3.10 __negative() . . . 120
5.23.3.11 __not() . . . 120
5.23.3.12 __string() . . . 120
5.23.3.13 __subtract() . . . 120
5.23.3.14 dump() . . . 121
5.23.3.15 is_equal() [1/6] . . . 121
5.23.3.16 is_equal() [2/6] . . . 122

5.23.3.17 is_equal() [3/6] . . . 123
5.23.3.18 is_equal() [4/6] . . . 123
5.23.3.19 is_equal() [5/6] . . . 124
5.23.3.20 is_equal() [6/6] . . . 124
5.23.3.21 makeCopy() . . . 124
5.24 Tang::ComputedExpressionError Class Reference . . . 125
5.24.1 Detailed Description . . . 126
5.24.2 Constructor & Destructor Documentation . . . 126
5.24.2.1 ComputedExpressionError() . . . 126
5.24.3 Member Function Documentation . . . 127
5.24.3.1 __add() . . . 127
5.24.3.2 __boolean() . . . 127
5.24.3.3 __divide() . . . 127
5.24.3.4 __equal() . . . 128
5.24.3.5 __float() . . . 128
5.24.3.6 __integer() . . . 129
5.24.3.7 __lessThan() . . . 129
5.24.3.8 __modulo() . . . 129
5.24.3.9 __multiply() . . . 130
5.24.3.10 __negative() . . . 130
5.24.3.11 __not() . . . 130
5.24.3.12 __string() . . . 131
5.24.3.13 __subtract() . . . 131
5.24.3.14 dump() . . . 131
5.24.3.15 is_equal() [1/6] . . . 131
5.24.3.16 is_equal() [2/6] . . . 132
5.24.3.17 is_equal() [3/6] . . . 132
5.24.3.18 is_equal() [4/6] . . . 133
5.24.3.19 is_equal() [5/6] . . . 133
5.24.3.20 is_equal() [6/6] . . . 133
5.24.3.21 makeCopy() . . . 134
5.25 Tang::ComputedExpressionFloat Class Reference . . . 134
5.25.1 Detailed Description . . . 136
5.25.2 Constructor & Destructor Documentation . . . 136
5.25.2.1 ComputedExpressionFloat() . . . 136
5.25.3 Member Function Documentation . . . 136
5.25.3.1 __add() . . . 136
5.25.3.2 __boolean() . . . 137
5.25.3.3 __divide() . . . 137
5.25.3.4 __equal() . . . 137
5.25.3.5 __float() . . . 138
5.25.3.6 __integer() . . . 138

5.25.3.7 __lessThan() . . . 138
5.25.3.8 __modulo() . . . 139
5.25.3.9 __multiply() . . . 139
5.25.3.10 __negative() . . . 140
5.25.3.11 __not() . . . 140
5.25.3.12 __string() . . . 140
5.25.3.13 __subtract() . . . 140
5.25.3.14 dump() . . . 141
5.25.3.15 is_equal() [1/6] . . . 141
5.25.3.16 is_equal() [2/6] . . . 141
5.25.3.17 is_equal() [3/6] . . . 142
5.25.3.18 is_equal() [4/6] . . . 142
5.25.3.19 is_equal() [5/6] . . . 143
5.25.3.20 is_equal() [6/6] . . . 143
5.25.3.21 makeCopy() . . . 143
5.26 Tang::ComputedExpressionInteger Class Reference . . . 144
5.26.1 Detailed Description . . . 145
5.26.2 Constructor & Destructor Documentation . . . 145
5.26.2.1 ComputedExpressionInteger() . . . 145
5.26.3 Member Function Documentation . . . 146
5.26.3.1 __add() . . . 146
5.26.3.2 __boolean() . . . 146
5.26.3.3 __divide() . . . 146
5.26.3.4 __equal() . . . 147
5.26.3.5 __float() . . . 147
5.26.3.6 __integer() . . . 148
5.26.3.7 __lessThan() . . . 148
5.26.3.8 __modulo() . . . 148
5.26.3.9 __multiply() . . . 149
5.26.3.10 __negative() . . . 149
5.26.3.11 __not() . . . 149
5.26.3.12 __string() . . . 150
5.26.3.13 __subtract() . . . 150
5.26.3.14 dump() . . . 150
5.26.3.15 is_equal() [1/6] . . . 151
5.26.3.16 is_equal() [2/6] . . . 151
5.26.3.17 is_equal() [3/6] . . . 151
5.26.3.18 is_equal() [4/6] . . . 152
5.26.3.19 is_equal() [5/6] . . . 152
5.26.3.20 is_equal() [6/6] . . . 153
5.26.3.21 makeCopy() . . . 153
5.27 Tang::ComputedExpressionString Class Reference . . . 153

5.27.1 Detailed Description . . . 155
5.27.2 Constructor & Destructor Documentation . . . 155
5.27.2.1 ComputedExpressionString() . . . 155
5.27.3 Member Function Documentation . . . 156
5.27.3.1 __add() . . . 156
5.27.3.2 __boolean() . . . 156
5.27.3.3 __divide() . . . 156
5.27.3.4 __equal() . . . 157
5.27.3.5 __float() . . . 157
5.27.3.6 __integer() . . . 157
5.27.3.7 __lessThan() . . . 158
5.27.3.8 __modulo() . . . 158
5.27.3.9 __multiply() . . . 158
5.27.3.10 __negative() . . . 159
5.27.3.11 __not() . . . 159
5.27.3.12 __string() . . . 159
5.27.3.13 __subtract() . . . 159
5.27.3.14 dump() . . . 160
5.27.3.15 is_equal() [1/6] . . . 160
5.27.3.16 is_equal() [2/6] . . . 160
5.27.3.17 is_equal() [3/6] . . . 161
5.27.3.18 is_equal() [4/6] . . . 161
5.27.3.19 is_equal() [5/6] . . . 162
5.27.3.20 is_equal() [6/6] . . . 162
5.27.3.21 makeCopy() . . . 162
5.28 Tang::Error Class Reference . . . 163
5.28.1 Detailed Description . . . 164
5.28.2 Constructor & Destructor Documentation . . . 164
5.28.2.1 Error() [1/2] . . . 164
5.28.2.2 Error() [2/2] . . . 164
5.28.3 Friends And Related Function Documentation . . . 164
5.28.3.1 operator<< . . . 165
5.29 Tang::GarbageCollected Class Reference . . . 165
5.29.1 Detailed Description . . . 167
5.29.2 Constructor & Destructor Documentation . . . 167
5.29.2.1 GarbageCollected() [1/3] . . . 167
5.29.2.2 GarbageCollected() [2/3] . . . 168
5.29.2.3 ~GarbageCollected() . . . 168
5.29.2.4 GarbageCollected() [3/3] . . . 168
5.29.3 Member Function Documentation . . . 168
5.29.3.1 make() . . . 168
5.29.3.2 operator"!() . . . 169

5.29.3.3 operator"!=() . . . 169
5.29.3.4 operator%() . . . 170
5.29.3.5 operator*() [1/2] . . . 171
5.29.3.6 operator*() [2/2] . . . 171
5.29.3.7 operator+() . . . 171
5.29.3.8 operator-() [1/2] . . . 172
5.29.3.9 operator-() [2/2] . . . 172
5.29.3.10 operator->() . . . 173
5.29.3.11 operator/() . . . 173
5.29.3.12 operator<() . . . 174
5.29.3.13 operator<=() . . . 174
5.29.3.14 operator=() [1/2] . . . 175
5.29.3.15 operator=() [2/2] . . . 175
5.29.3.16 operator==() [1/8] . . . 176
5.29.3.17 operator==() [2/8] . . . 176
5.29.3.18 operator==() [3/8] . . . 177
5.29.3.19 operator==() [4/8] . . . 177
5.29.3.20 operator==() [5/8] . . . 177
5.29.3.21 operator==() [6/8] . . . 178
5.29.3.22 operator==() [7/8] . . . 178
5.29.3.23 operator==() [8/8] . . . 178
5.29.3.24 operator>() . . . 180
5.29.3.25 operator>=() . . . 180
5.29.4 Friends And Related Function Documentation . . . 181
5.29.4.1 operator<< . . . 181
5.30 Tang::location Class Reference . . . 181
5.30.1 Detailed Description . . . 183
5.31 Tang::position Class Reference . . . 183
5.31.1 Detailed Description . . . 184
5.32 Tang::Program Class Reference . . . 184
5.32.1 Detailed Description . . . 186
5.32.2 Member Enumeration Documentation . . . 186
5.32.2.1 CodeType . . . 186
5.32.3 Constructor & Destructor Documentation . . . 186
5.32.3.1 Program() . . . 186
5.32.4 Member Function Documentation . . . 186
5.32.4.1 addBytecode() . . . 186
5.32.4.2 addIdentifier() . . . 188
5.32.4.3 addString() . . . 188
5.32.4.4 dumpBytecode() . . . 188
5.32.4.5 execute() . . . 189
5.32.4.6 getAst() . . . 189

5.32.4.7 getBytecode() . . . 189
5.32.4.8 getCode() . . . 190
5.32.4.9 getIdentifiers() . . . 190
5.32.4.10 getResult() . . . 190
5.32.4.11 getStrings() . . . 190
5.32.4.12 pushEnvironment() . . . 190
5.32.4.13 setFunctionStackDeclaration() . . . 191
5.32.4.14 setJumpTarget() . . . 191
5.32.5 Member Data Documentation . . . 191
5.32.5.1 functionsDeclared . . . 191
5.33 Tang::SingletonObjectPool< T > Class Template Reference . . . 192
5.33.1 Detailed Description . . . 192
5.33.2 Member Function Documentation . . . 192
5.33.2.1 get() . . . 193
5.33.2.2 getInstance() . . . 193
5.33.2.3 recycle() . . . 193
5.34 Tang::TangBase Class Reference . . . 193
5.34.1 Detailed Description . . . 194
5.34.2 Constructor & Destructor Documentation . . . 194
5.34.2.1 TangBase() . . . 194
5.34.3 Member Function Documentation . . . 194
5.34.3.1 compileScript() . . . 194
5.35 Tang::TangScanner Class Reference . . . 195
5.35.1 Detailed Description . . . 196
5.35.2 Constructor & Destructor Documentation . . . 196
5.35.2.1 TangScanner() . . . 196
5.35.3 Member Function Documentation . . . 196
5.35.3.1 get_next_token() . . . 196

6 File Documentation **197**

6.1 build/generated/location.hh File Reference . . . 197
6.1.1 Detailed Description . . . 198
6.1.2 Function Documentation . . . 198
6.1.2.1 operator<<() [1/2] . . . 198
6.1.2.2 operator<<() [2/2] . . . 199
6.2 include/astNode.hpp File Reference . . . 199
6.2.1 Detailed Description . . . 200
6.3 include/astNodeAssign.hpp File Reference . . . 200
6.3.1 Detailed Description . . . 201
6.4 include/astNodeBinary.hpp File Reference . . . 201
6.4.1 Detailed Description . . . 202
6.5 include/astNodeBlock.hpp File Reference . . . 202

6.5.1 Detailed Description . . . 203
6.6 include/astNodeBoolean.hpp File Reference . . . 203
6.6.1 Detailed Description . . . 204
6.7 include/astNodeCast.hpp File Reference . . . 204
6.7.1 Detailed Description . . . 205
6.8 include/astNodeDoWhile.hpp File Reference . . . 205
6.8.1 Detailed Description . . . 206
6.9 include/astNodeFloat.hpp File Reference . . . 206
6.9.1 Detailed Description . . . 207
6.10 include/astNodeFor.hpp File Reference . . . 207
6.10.1 Detailed Description . . . 208
6.11 include/astNodeFunctionCall.hpp File Reference . . . 208
6.11.1 Detailed Description . . . 209
6.12 include/astNodeFunctionDeclaration.hpp File Reference . . . 209
6.12.1 Detailed Description . . . 210
6.13 include/astNodeIdentifier.hpp File Reference . . . 210
6.13.1 Detailed Description . . . 211
6.14 include/astNodeIfElse.hpp File Reference . . . 211
6.14.1 Detailed Description . . . 212
6.15 include/astNodeInteger.hpp File Reference . . . 212
6.15.1 Detailed Description . . . 213
6.16 include/astNodePrint.hpp File Reference . . . 213
6.16.1 Detailed Description . . . 214
6.17 include/astNodeReturn.hpp File Reference . . . 214
6.17.1 Detailed Description . . . 215
6.18 include/astNodeString.hpp File Reference . . . 215
6.18.1 Detailed Description . . . 216
6.19 include/astNodeTernary.hpp File Reference . . . 216
6.19.1 Detailed Description . . . 217
6.20 include/astNodeUnary.hpp File Reference . . . 217
6.20.1 Detailed Description . . . 218
6.21 include/astNodeWhile.hpp File Reference . . . 218
6.21.1 Detailed Description . . . 219
6.22 include/computedExpression.hpp File Reference . . . 219
6.22.1 Detailed Description . . . 219
6.23 include/computedExpressionBoolean.hpp File Reference . . . 220
6.23.1 Detailed Description . . . 220
6.24 include/computedExpressionCompiledFunction.hpp File Reference . . . 221
6.24.1 Detailed Description . . . 221
6.25 include/computedExpressionError.hpp File Reference . . . 222
6.25.1 Detailed Description . . . 222
6.26 include/computedExpressionFloat.hpp File Reference . . . 223

6.26.1 Detailed Description . . . 223
6.27 include/computedExpressionInteger.hpp File Reference . . . 224
6.27.1 Detailed Description . . . 224
6.28 include/computedExpressionString.hpp File Reference . . . 225
6.28.1 Detailed Description . . . 225
6.29 include/error.hpp File Reference . . . 226
6.29.1 Detailed Description . . . 226
6.30 include/garbageCollected.hpp File Reference . . . 227
6.30.1 Detailed Description . . . 227
6.31 include/macros.hpp File Reference . . . 227
6.31.1 Detailed Description . . . 228
6.32 include/opcode.hpp File Reference . . . 228
6.32.1 Detailed Description . . . 228
6.32.2 Enumeration Type Documentation . . . 228
6.32.2.1 Opcode . . . 228
6.33 include/program.hpp File Reference . . . 229
6.33.1 Detailed Description . . . 230
6.34 include/singletonObjectPool.hpp File Reference . . . 230
6.34.1 Detailed Description . . . 231
6.35 include/tang.hpp File Reference . . . 231
6.35.1 Detailed Description . . . 232
6.36 include/tangBase.hpp File Reference . . . 232
6.36.1 Detailed Description . . . 233
6.37 include/tangScanner.hpp File Reference . . . 234
6.37.1 Detailed Description . . . 235
6.38 src/astNode.cpp File Reference . . . 235
6.38.1 Detailed Description . . . 235
6.39 src/astNodeAssign.cpp File Reference . . . 235
6.39.1 Detailed Description . . . 236
6.40 src/astNodeBinary.cpp File Reference . . . 236
6.40.1 Detailed Description . . . 237
6.41 src/astNodeBlock.cpp File Reference . . . 237
6.41.1 Detailed Description . . . 237
6.42 src/astNodeBoolean.cpp File Reference . . . 237
6.42.1 Detailed Description . . . 238
6.43 src/astNodeCast.cpp File Reference . . . 238
6.43.1 Detailed Description . . . 239
6.44 src/astNodeDoWhile.cpp File Reference . . . 239
6.44.1 Detailed Description . . . 239
6.45 src/astNodeFloat.cpp File Reference . . . 239
6.45.1 Detailed Description . . . 240
6.46 src/astNodeFor.cpp File Reference . . . 240

6.46.1 Detailed Description . . . 241
6.47 src/astNodeFunctionCall.cpp File Reference . . . 241
6.47.1 Detailed Description . . . 242
6.48 src/astNodeFunctionDeclaration.cpp File Reference . . . 242
6.48.1 Detailed Description . . . 243
6.49 src/astNodeIdentifier.cpp File Reference . . . 243
6.49.1 Detailed Description . . . 244
6.50 src/astNodeIfElse.cpp File Reference . . . 244
6.50.1 Detailed Description . . . 244
6.51 src/astNodeInteger.cpp File Reference . . . 244
6.51.1 Detailed Description . . . 245
6.52 src/astNodePrint.cpp File Reference . . . 245
6.52.1 Detailed Description . . . 246
6.53 src/astNodeReturn.cpp File Reference . . . 246
6.53.1 Detailed Description . . . 246
6.54 src/astNodeString.cpp File Reference . . . 246
6.54.1 Detailed Description . . . 247
6.55 src/astNodeTernary.cpp File Reference . . . 247
6.55.1 Detailed Description . . . 248
6.56 src/astNodeUnary.cpp File Reference . . . 248
6.56.1 Detailed Description . . . 249
6.57 src/astNodeWhile.cpp File Reference . . . 249
6.57.1 Detailed Description . . . 250
6.58 src/computedExpression.cpp File Reference . . . 250
6.58.1 Detailed Description . . . 251
6.59 src/computedExpressionBoolean.cpp File Reference . . . 251
6.59.1 Detailed Description . . . 252
6.60 src/computedExpressionCompiledFunction.cpp File Reference . . . 252
6.60.1 Detailed Description . . . 252
6.61 src/computedExpressionError.cpp File Reference . . . 252
6.61.1 Detailed Description . . . 253
6.62 src/computedExpressionFloat.cpp File Reference . . . 253
6.62.1 Detailed Description . . . 254
6.63 src/computedExpressionInteger.cpp File Reference . . . 254
6.63.1 Detailed Description . . . 254
6.64 src/computedExpressionString.cpp File Reference . . . 255
6.64.1 Detailed Description . . . 255
6.65 src/error.cpp File Reference . . . 255
6.65.1 Detailed Description . . . 256
6.65.2 Function Documentation . . . 256
6.65.2.1 operator<<() . . . 256
6.66 src/program-dumpBytecode.cpp File Reference . . . 257

6.66.1 Detailed Description . . . 257
6.66.2 Macro Definition Documentation . . . 257
6.66.2.1 DUMPPROGRAMCHECK . . . 258
6.67 src/program-execute.cpp File Reference . . . 258
6.67.1 Detailed Description . . . 259
6.67.2 Macro Definition Documentation . . . 259
6.67.2.1 EXECUTEPROGRAMCHECK . . . 259
6.67.2.2 STACKCHECK . . . 259
6.68 src/program.cpp File Reference . . . 259
6.68.1 Detailed Description . . . 260
6.69 src/tangBase.cpp File Reference . . . 260
6.69.1 Detailed Description . . . 261
6.70 test/test.cpp File Reference . . . 261
6.70.1 Detailed Description . . . 262
6.71 test/testGarbageCollected.cpp File Reference . . . 262
6.71.1 Detailed Description . . . 263
6.72 test/testSingletonObjectPool.cpp File Reference . . . 263
6.72.1 Detailed Description . . . 263

Index **265**

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 . . . 16
- Tang::AstNodeBinary . . . 20
- Tang::AstNodeBlock . . . 25
- Tang::AstNodeBoolean . . . 28
- Tang::AstNodeCast . . . 32
- Tang::AstNodeDoWhile . . . 36
- Tang::AstNodeFloat . . . 40
- Tang::AstNodeFor . . . 44
- Tang::AstNodeFunctionCall . . . 49
- Tang::AstNodeFunctionDeclaration . . . 53
- Tang::AstNodeIdentifier . . . 57
- Tang::AstNodeIfElse . . . 61
- Tang::AstNodeInteger . . . 67
- Tang::AstNodePrint . . . 70
- Tang::AstNodeReturn . . . 74
- Tang::AstNodeString . . . 78
- Tang::AstNodeTernary . . . 83
- Tang::AstNodeUnary . . . 86
- Tang::AstNodeWhile . . . 91

Tang::ComputedExpression . . . 95
- Tang::ComputedExpressionBoolean . . . 105
- Tang::ComputedExpressionCompiledFunction . . . 114
- Tang::ComputedExpressionError . . . 125
- Tang::ComputedExpressionFloat . . . 134
- Tang::ComputedExpressionInteger . . . 144
- Tang::ComputedExpressionString . . . 153

Tang::Error . . . 163
Tang::GarbageCollected . . . 165
Tang::location . . . 181
Tang::position . . . 183
Tang::Program . . . 184
Tang::SingletonObjectPool< T > . . . 192
Tang::TangBase . . . 193
TangTangFlexLexer
- Tang::TangScanner . . . 195

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 . . . 16
Tang::AstNodeBinary
An AstNode that represents a binary expression . . . 20
Tang::AstNodeBlock
An AstNode that represents a code block . . . 25
Tang::AstNodeBoolean
An AstNode that represents a boolean literal . . . 28
Tang::AstNodeCast
An AstNode that represents a typecast of an expression . . . 32
Tang::AstNodeDoWhile
An AstNode that represents a do..while statement . . . 36
Tang::AstNodeFloat
An AstNode that represents an float literal . . . 40
Tang::AstNodeFor
An AstNode that represents an if() statement . . . 44
Tang::AstNodeFunctionCall
An AstNode that represents a function call . . . 49
Tang::AstNodeFunctionDeclaration
An AstNode that represents a function declaration . . . 53
Tang::AstNodeIdentifier
An AstNode that represents an identifier . . . 57
Tang::AstNodeIfElse
An AstNode that represents an if..else statement . . . 61
Tang::AstNodeInteger
An AstNode that represents an integer literal . . . 67
Tang::AstNodePrint
An AstNode that represents a print typeeration . . . 70
Tang::AstNodeReturn
An AstNode that represents a `return` statement . . . 74
Tang::AstNodeString
An AstNode that represents a string literal . . . 78
Tang::AstNodeTernary
An AstNode that represents a ternary expression . . . 83

Tang::AstNodeUnary
An AstNode that represents a unary negation 86
Tang::AstNodeWhile
An AstNode that represents a while statement 91
Tang::ComputedExpression
Represents the result of a computation that has been executed 95
Tang::ComputedExpressionBoolean
Represents an Boolean that is the result of a computation 105
Tang::ComputedExpressionCompiledFunction
Represents a Compiled Function declared in the script 114
Tang::ComputedExpressionError
Represents a Runtime Error 125
Tang::ComputedExpressionFloat
Represents a Float that is the result of a computation 134
Tang::ComputedExpressionInteger
Represents an Integer that is the result of a computation 144
Tang::ComputedExpressionString
Represents a String that is the result of a computation 153
Tang::Error
Used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error 163
Tang::GarbageCollected
A container that acts as a resource-counting garbage collector for the specified type 165
Tang::location
Two points in a source file 181
Tang::position
A point in a source file 183
Tang::Program
Represents a compiled script or template that may be executed 184
Tang::SingletonObjectPool< T >
A thread-safe, singleton object pool of the designated type 192
Tang::TangBase
The base class for the Tang programming language 193
Tang::TangScanner
The Flex lexer class for the main Tang language 195

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 . . . 197
include/astNode.hpp
Declare the Tang::AstNode base class . . . 199
include/astNodeAssign.hpp
Declare the Tang::AstNodeAssign class . . . 200
include/astNodeBinary.hpp
Declare the Tang::AstNodeBinary class . . . 201
include/astNodeBlock.hpp
Declare the Tang::AstNodeBlock class . . . 202
include/astNodeBoolean.hpp
Declare the Tang::AstNodeBoolean class . . . 203
include/astNodeCast.hpp
Declare the Tang::AstNodeCast class . . . 204
include/astNodeDoWhile.hpp
Declare the Tang::AstNodeDoWhile class . . . 205
include/astNodeFloat.hpp
Declare the Tang::AstNodeFloat class . . . 206
include/astNodeFor.hpp
Declare the Tang::AstNodeFor class . . . 207
include/astNodeFunctionCall.hpp
Declare the Tang::AstNodeFunctionCall class . . . 208
include/astNodeFunctionDeclaration.hpp
Declare the Tang::AstNodeFunctionDeclaration class . . . 209
include/astNodeIdentifier.hpp
Declare the Tang::AstNodeIdentifier class . . . 210
include/astNodeIfElse.hpp
Declare the Tang::AstNodeIfElse class . . . 211
include/astNodeInteger.hpp
Declare the Tang::AstNodeInteger class . . . 212
include/astNodePrint.hpp
Declare the Tang::AstNodePrint class . . . 213
include/astNodeReturn.hpp
Declare the Tang::AstNodeReturn class . . . 214
include/astNodeString.hpp
Declare the Tang::AstNodeString class . . . 215

include/astNodeTernary.hpp
Declare the Tang::AstNodeTernary class . . . 216
include/astNodeUnary.hpp
Declare the Tang::AstNodeUnary class . . . 217
include/astNodeWhile.hpp
Declare the Tang::AstNodeWhile class . . . 218
include/computedExpression.hpp
Declare the Tang::ComputedExpression base class . . . 219
include/computedExpressionBoolean.hpp
Declare the Tang::ComputedExpressionBoolean class . . . 220
include/computedExpressionCompiledFunction.hpp
Declare the Tang::ComputedExpressionCompiledFunction class . . . 221
include/computedExpressionError.hpp
Declare the Tang::ComputedExpressionError class . . . 222
include/computedExpressionFloat.hpp
Declare the Tang::ComputedExpressionFloat class . . . 223
include/computedExpressionInteger.hpp
Declare the Tang::ComputedExpressionInteger class . . . 224
include/computedExpressionString.hpp
Declare the Tang::ComputedExpressionString class . . . 225
include/error.hpp
Declare the Tang::Error class used to describe syntax and runtime errors . . . 226
include/garbageCollected.hpp
Declare the Tang::GarbageCollected class . . . 227
include/macros.hpp
Contains generic macros . . . 227
include/opcode.hpp
Declare the Opcodes used in the Bytecode representation of a program . . . 228
include/program.hpp
Declare the Tang::Program class used to compile and execute source code . . . 229
include/singletonObjectPool.hpp
Declare the Tang::SingletonObjectPool class . . . 230
include/tang.hpp
Header file supplied for use by 3rd party code so that they can easily include all necessary headers . . . 231
include/tangBase.hpp
Declare the Tang::TangBase class used to interact with Tang . . . 232
include/tangScanner.hpp
Declare the Tang::TangScanner used to tokenize a Tang script . . . 234
src/astNode.cpp
Define the Tang::AstNode class . . . 235
src/astNodeAssign.cpp
Define the Tang::AstNodeAssign class . . . 235
src/astNodeBinary.cpp
Define the Tang::AstNodeBinary class . . . 236
src/astNodeBlock.cpp
Define the Tang::AstNodeBlock class . . . 237
src/astNodeBoolean.cpp
Define the Tang::AstNodeBoolean class . . . 237
src/astNodeCast.cpp
Define the Tang::AstNodeCast class . . . 238
src/astNodeDoWhile.cpp
Define the Tang::AstNodeDoWhile class . . . 239
src/astNodeFloat.cpp
Define the Tang::AstNodeFloat class . . . 239
src/astNodeFor.cpp
Define the Tang::AstNodeFor class . . . 240

src/astNodeFunctionCall.cpp
Define the Tang::AstNodeFunctionCall class . . . 241
src/astNodeFunctionDeclaration.cpp
Define the Tang::AstNodeFunctionDeclaration class . . . 242
src/astNodeIdentifier.cpp
Define the Tang::AstNodeIdentifier class . . . 243
src/astNodeIfElse.cpp
Define the Tang::AstNodeIfElse class . . . 244
src/astNodeInteger.cpp
Define the Tang::AstNodeInteger class . . . 244
src/astNodePrint.cpp
Define the Tang::AstNodePrint class . . . 245
src/astNodeReturn.cpp
Define the Tang::AstNodeReturn class . . . 246
src/astNodeString.cpp
Define the Tang::AstNodeString class . . . 246
src/astNodeTernary.cpp
Define the Tang::AstNodeTernary class . . . 247
src/astNodeUnary.cpp
Define the Tang::AstNodeUnary class . . . 248
src/astNodeWhile.cpp
Define the Tang::AstNodeWhile class . . . 249
src/computedExpression.cpp
Define the Tang::ComputedExpression class . . . 250
src/computedExpressionBoolean.cpp
Define the Tang::ComputedExpressionBoolean class . . . 251
src/computedExpressionCompiledFunction.cpp
Define the Tang::ComputedExpressionCompiledFunction class . . . 252
src/computedExpressionError.cpp
Define the Tang::ComputedExpressionError class . . . 252
src/computedExpressionFloat.cpp
Define the Tang::ComputedExpressionFloat class . . . 253
src/computedExpressionInteger.cpp
Define the Tang::ComputedExpressionInteger class . . . 254
src/computedExpressionString.cpp
Define the Tang::ComputedExpressionString class . . . 255
src/error.cpp
Define the Tang::Error class . . . 255
src/program-dumpBytecode.cpp
Define the Tang::Program::dumpBytecode method . . . 257
src/program-execute.cpp
Define the Tang::Program::execute method . . . 258
src/program.cpp
Define the Tang::Program class . . . 259
src/tangBase.cpp
Define the Tang::TangBase class . . . 260
test/test.cpp
Test the general language behaviors . . . 261
test/testGarbageCollected.cpp
Test the generic behavior of the Tang::GarbageCollected class . . . 262
test/testSingletonObjectPool.cpp
Test the generic behavior of the Tang::SingletonObjectPool class . . . 263

Chapter 5

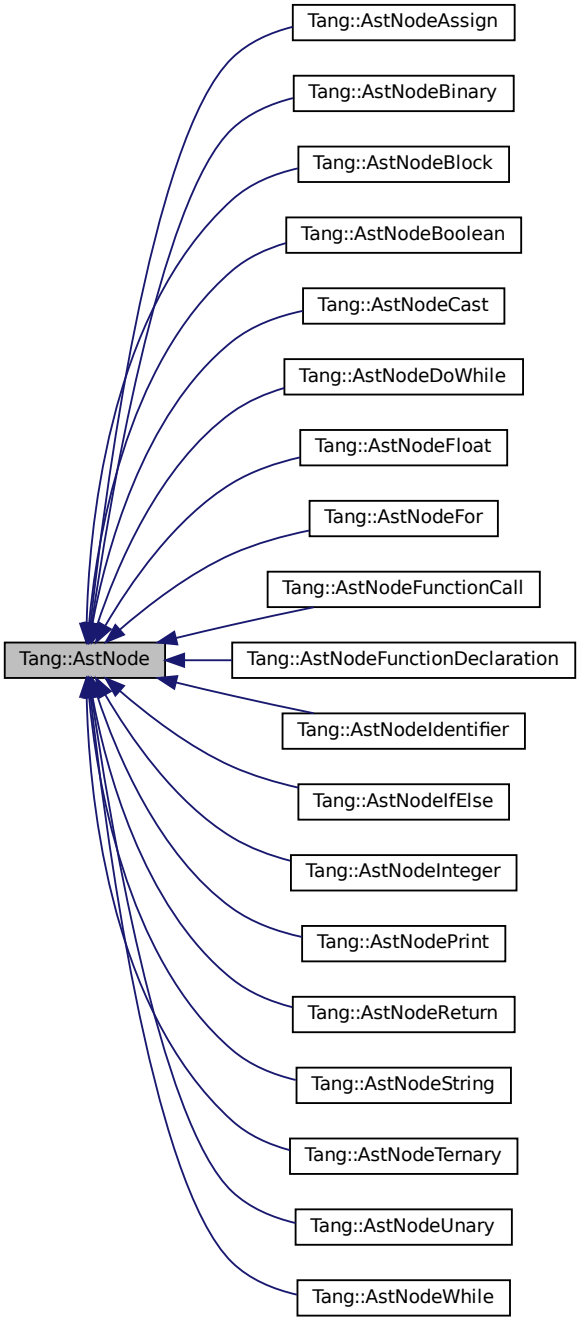
Class Documentation

5.1 Tang::AstNode Class Reference

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

`#include <astNode.hpp>`

Inheritance diagram for Tang::AstNode:



Public Member Functions

- AstNode (Tang::location location)

  The generic constructor.
- virtual ~AstNode ()

  The object destructor.
- virtual std::string dump (std::string indent="") const

Return a string that describes the contents of the node.

- virtual void compile (Tang::Program &program) const

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

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

5.1.1 Detailed Description

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

By default, it will represent a NULL value. There will be *many* derived classes, each one conveying the syntactic meaning of the code that it represents.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AstNode()

```
AstNode::AstNode (
            Tang::location location )
```

The generic constructor.

It should never be called on its own.

Parameters

location	The location associated with this node.

5.1.3 Member Function Documentation

5.1.3.1 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.1.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

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

5.1.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

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

5.1.3.4 compile()

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

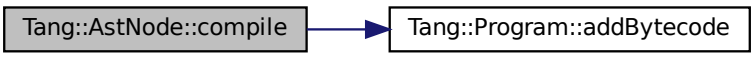
Compile the ast of the provided Tang::Program.

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:



5.1.3.5 dump()

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

Return a string that describes the contents of the node.

Parameters

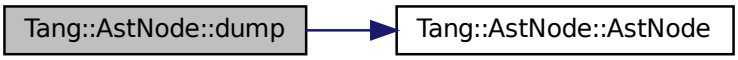
indent	A string used to indent the dump.

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:

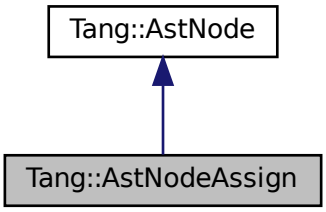
- include/astNode.hpp
- src/astNode.cpp

5.2 Tang::AstNodeAssign Class Reference

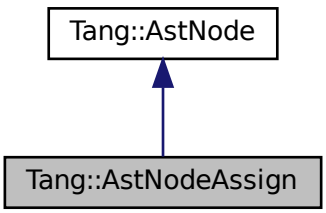
An AstNode that represents a binary expression.

`#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 collectIdentifiers (Program &program) const override

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

5.2.1 Detailed Description

An AstNode that represents a binary expression.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AstNodeAssign()

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

The constructor.

Parameters

lhs	The left hand side expression.
rhs	The right hand side expression.
location	The location associated with the expression.

5.2.3 Member Function Documentation

5.2.3.1 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.2.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.2.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.2.3.4 compile()

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

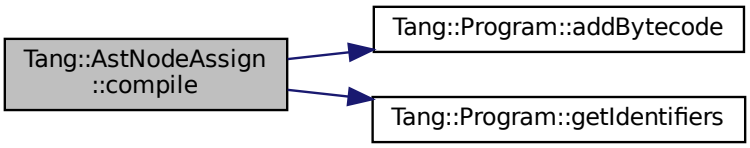
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.2.3.5 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

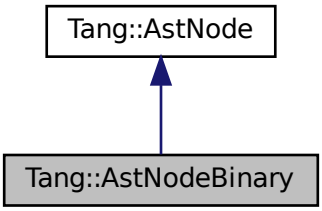
- include/astNodeAssign.hpp
- src/astNodeAssign.cpp

5.3 Tang::AstNodeBinary Class Reference

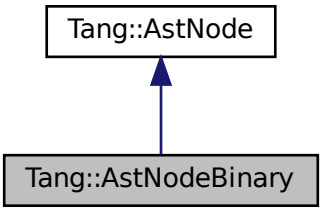
An AstNode that represents a binary expression.

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

Inheritance diagram for Tang::AstNodeBinary:



Collaboration diagram for Tang::AstNodeBinary:



Public Types

- enum Operation {
  Add , Subtract , Multiply , Divide ,
  Modulo , LessThan , LessThanEqual , GreaterThan ,
  GreaterThanEqual , Equal , NotEqual , And ,
  Or }

  Indicates the type of binary expression that this node represents.

Public Member Functions

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

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

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

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

Compile a list of all variables in the scope.

- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

5.3.1 Detailed Description

An AstNode that represents a binary expression.

5.3.2 Member Enumeration Documentation

5.3.2.1 Operation

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

Indicates the type of binary expression that this node represents.

Enumerator

Add	Indicates lhs + rhs.
Subtract	Indicates lhs - rhs.
Multiply	Indicates lhs $*$ rhs.
Divide	Indicates lhs / rhs.
Modulo	Indicates lhs % rhs.
LessThan	Indicates lhs $<$ rhs.
LessThanEqual	Indicates lhs $<=$ rhs.
GreaterThan	Indicates lhs $>$ rhs.
GreaterThanEqual	Indicates lhs $>=$ rhs.
Equal	Indicates lhs == rhs.
NotEqual	Indicates lhs != rhs.
And	Indicates lhs && rhs with short-circuit evaluation.
Or	Indicates lhs $\|\|$ rhs with short-circuit evaluation.

5.3.3 Constructor & Destructor Documentation

5.3.3.1 AstNodeBinary()

```
AstNodeBinary::AstNodeBinary (
            Operation op,
```

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

The constructor.

Parameters

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

5.3.4 Member Function Documentation

5.3.4.1 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.3.4.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.3.4.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.3.4.4 compile()

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

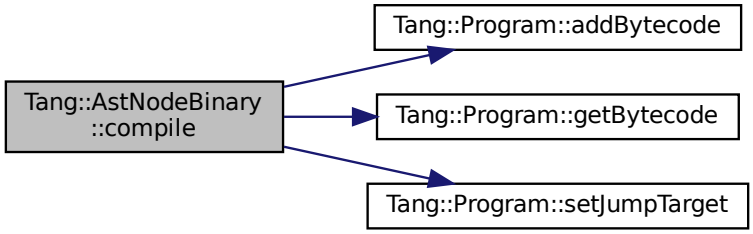
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.3.4.5 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

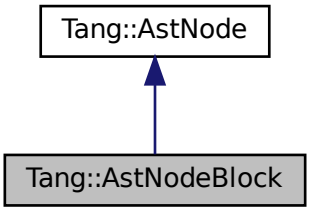
- include/astNodeBinary.hpp
- src/astNodeBinary.cpp

5.4 Tang::AstNodeBlock Class Reference

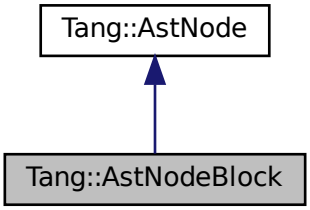
An AstNode that represents a code block.

`#include <astNodeBlock.hpp>`

Inheritance diagram for Tang::AstNodeBlock:



Collaboration diagram for Tang::AstNodeBlock:



Public Member Functions

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

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

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

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

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const override

  Compile a list of the names of functions that are declared in this scope.

5.4.1 Detailed Description

An AstNode that represents a code block.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 AstNodeBlock()

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

The constructor.

Parameters

statements	The statements of the code block.
location	The location associated with the expression.

5.4.3 Member Function Documentation

5.4.3.1 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.4.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.4.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.4.3.4 compile()

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

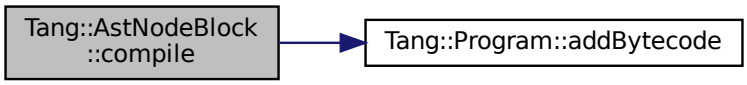
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.4.3.5 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

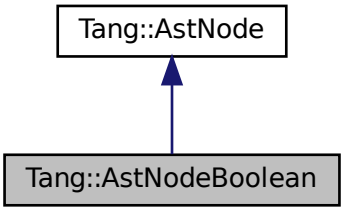
- include/astNodeBlock.hpp
- src/astNodeBlock.cpp

5.5 Tang::AstNodeBoolean Class Reference

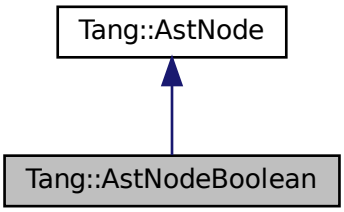
An AstNode that represents a boolean literal.

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

Inheritance diagram for Tang::AstNodeBoolean:



Collaboration diagram for Tang::AstNodeBoolean:



Public Member Functions

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

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

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

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

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

5.5.1 Detailed Description

An AstNode that represents a boolean literal.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 AstNodeBoolean()

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

The constructor.

Parameters

val	The boolean to represent.
location	The location associated with the expression.

5.5.3 Member Function Documentation

5.5.3.1 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.5.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

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

5.5.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

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

5.5.3.4 compile()

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

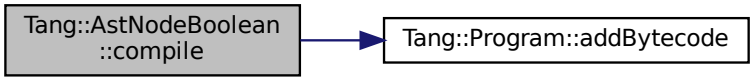
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.5.3.5 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

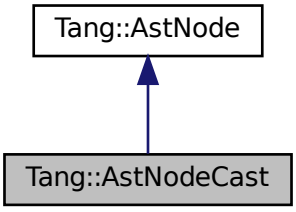
- include/astNodeBoolean.hpp
- src/astNodeBoolean.cpp

5.6 Tang::AstNodeCast Class Reference

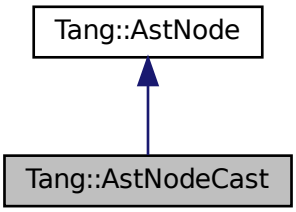
An AstNode that represents a typecast of an expression.

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

Inheritance diagram for Tang::AstNodeCast:



Collaboration diagram for Tang::AstNodeCast:



Public Types

- enum Type { Integer , Float , Boolean }

  The possible types that can be cast to.

Public Member Functions

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

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

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

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

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

5.6.1 Detailed Description

An AstNode that represents a typecast of an expression.

5.6.2 Member Enumeration Documentation

5.6.2.1 Type

`enum Tang::AstNodeCast::Type`

The possible types that can be cast to.

Enumerator

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

5.6.3 Constructor & Destructor Documentation

5.6.3.1 AstNodeCast()

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

The constructor.

Parameters

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

5.6.4 Member Function Documentation

5.6.4.1 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.6.4.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

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

5.6.4.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

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

5.6.4.4 compile()

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

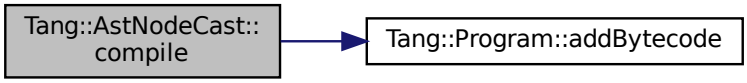
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.6.4.5 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

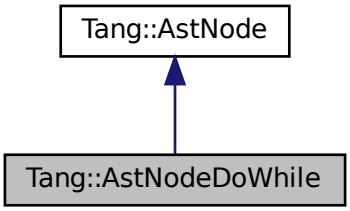
- include/astNodeCast.hpp
- src/astNodeCast.cpp

5.7 Tang::AstNodeDoWhile Class Reference

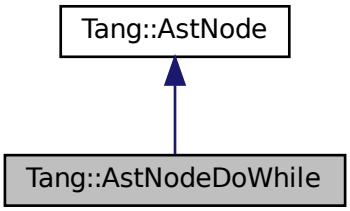
An AstNode that represents a do..while statement.

```
#include <astNodeDoWhile.hpp>
```

Inheritance diagram for Tang::AstNodeDoWhile:



Collaboration diagram for Tang::AstNodeDoWhile:



Public Member Functions

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

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

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

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

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

5.7.1 Detailed Description

An AstNode that represents a do..while statement.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 AstNodeDoWhile()

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

The constructor.

Parameters

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

5.7.3 Member Function Documentation

5.7.3.1 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.7.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.7.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.7.3.4 compile()

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

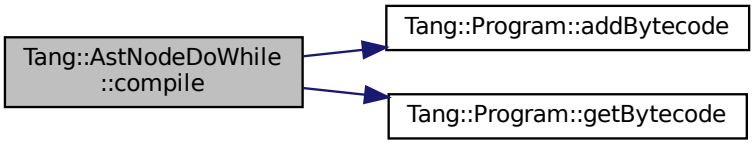
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.7.3.5 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:

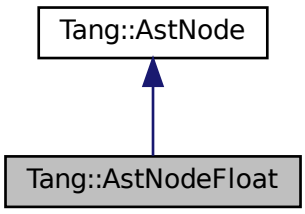
- include/astNodeDoWhile.hpp
- src/astNodeDoWhile.cpp

5.8 Tang::AstNodeFloat Class Reference

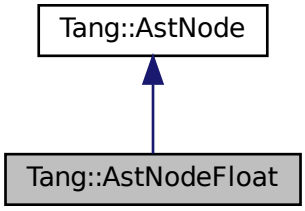
An AstNode that represents an float literal.

```
#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 collectIdentifiers (Program &program) const
  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const
  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const
  Compile a list of the names of functions that are declared in this scope.

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 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.8.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

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

5.8.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

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

5.8.3.4 compile()

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

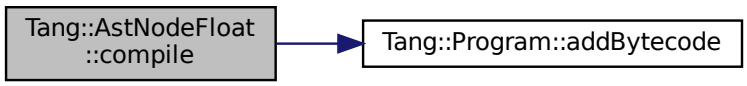
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.8.3.5 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

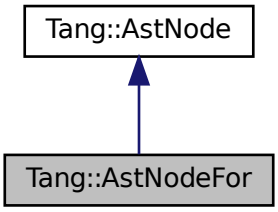
- include/astNodeFloat.hpp
- src/astNodeFloat.cpp

5.9 Tang::AstNodeFor Class Reference

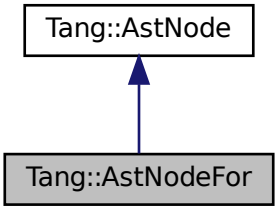
An AstNode that represents an if() statement.

`#include <astNodeFor.hpp>`

Inheritance diagram for Tang::AstNodeFor:



Collaboration diagram for Tang::AstNodeFor:



Public Member Functions

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

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

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

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

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

5.9.1 Detailed Description

An AstNode that represents an if() statement.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 AstNodeFor()

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

The constructor.

Parameters

initialization	The expression to be executed first.
condition	The expression which determines whether the codeBlock is executed.
increment	The expression to be executed after each codeBlock.
codeBlock	The statement executed when the condition is true.
location	The location associated with the expression.

5.9.3 Member Function Documentation

5.9.3.1 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.9.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.9.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.9.3.4 compile()

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

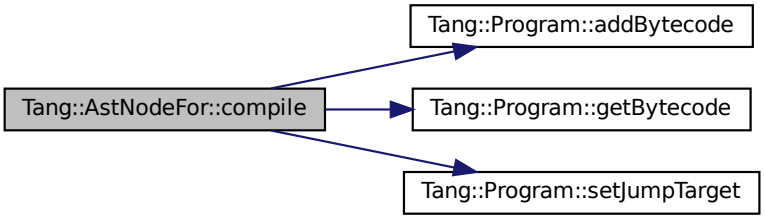
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.9.3.5 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:

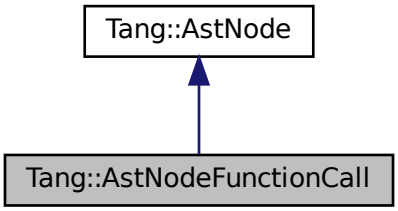
- include/astNodeFor.hpp
- src/astNodeFor.cpp

5.10 Tang::AstNodeFunctionCall Class Reference

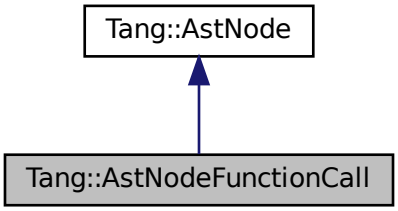
An AstNode that represents a function call.

`#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 collectIdentifiers (Program &program) const override

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

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 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.10.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.10.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.10.3.4 compile()

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

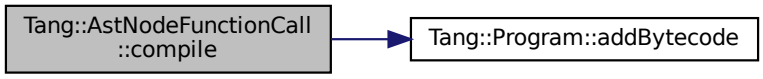
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.10.3.5 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:

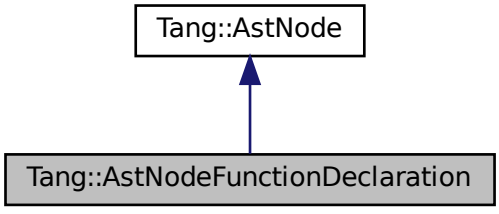
- include/astNodeFunctionCall.hpp
- src/astNodeFunctionCall.cpp

5.11 Tang::AstNodeFunctionDeclaration Class Reference

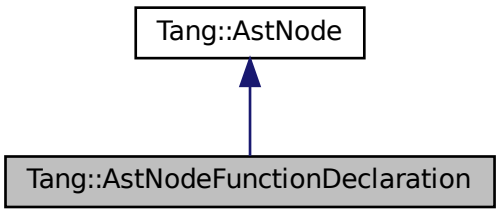
An AstNode that represents a function declaration.

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

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

  Compile a list of the names of functions that are declared in this scope.
- virtual void collectStrings (Program &program) const

  Compile a list of all string constants in the scope.

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 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.11.3.2 collectIdentifiers()

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

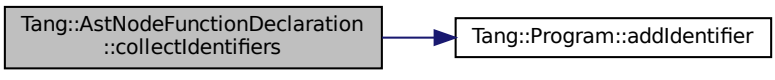
Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.11.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

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

5.11.3.4 compile()

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

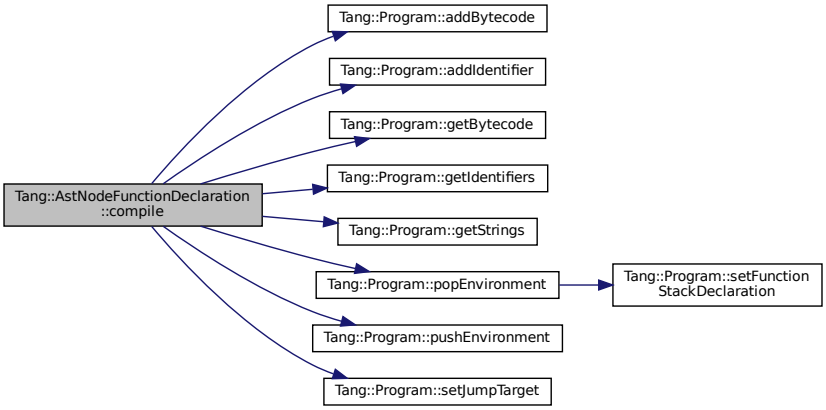
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.11.3.5 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:

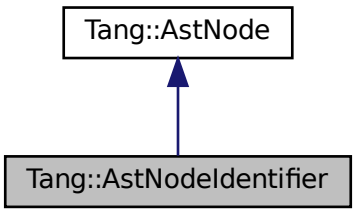
- include/astNodeFunctionDeclaration.hpp
- src/astNodeFunctionDeclaration.cpp

5.12 Tang::AstNodeIdentifier Class Reference

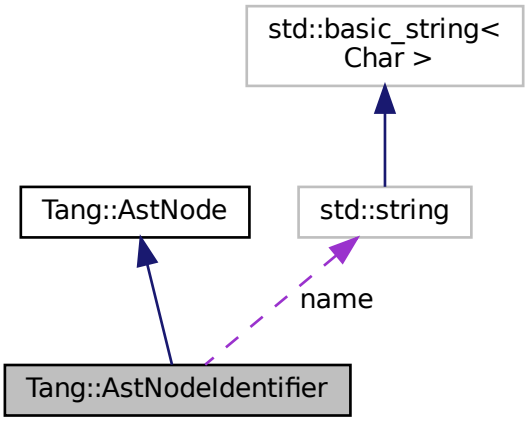
An AstNode that represents an identifier.

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

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

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 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.12.3.2 collectIdentifiers()

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

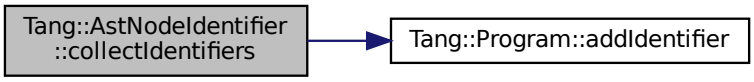
Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.12.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

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

5.12.3.4 compile()

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

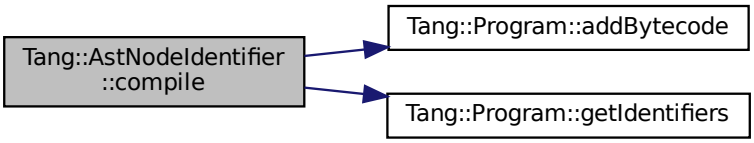
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.12.3.5 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:

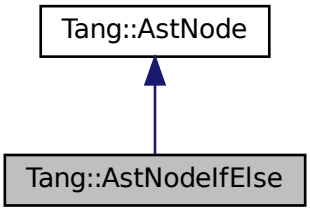
- include/astNodeIdentifier.hpp
- src/astNodeIdentifier.cpp

5.13 Tang::AstNodeIfElse Class Reference

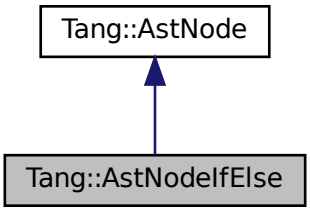
An AstNode that represents an if..else statement.

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

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

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 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.13.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.13.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.13.3.4 compile()

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

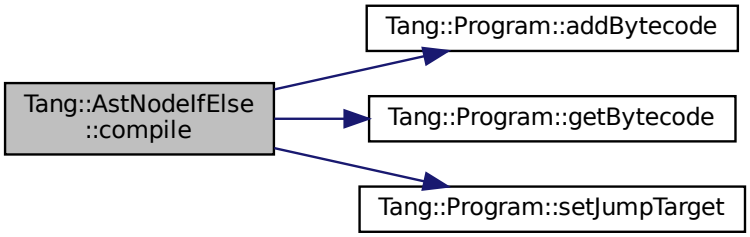
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.13.3.5 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:

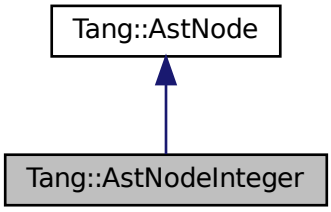
- include/astNodeIfElse.hpp
- src/astNodeIfElse.cpp

5.14 Tang::AstNodeInteger Class Reference

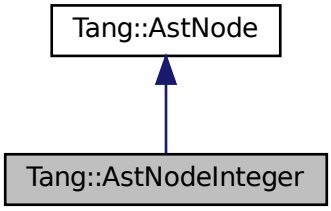
An AstNode that represents an integer literal.

`#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 collectIdentifiers (Program &program) const
  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const
  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const
  Compile a list of the names of functions that are declared in this scope.

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 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.14.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

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

5.14.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

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

5.14.3.4 compile()

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

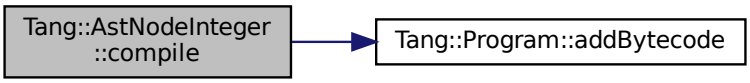
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.14.3.5 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:

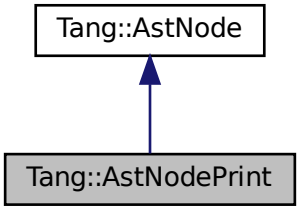
- include/astNodeInteger.hpp
- src/astNodeInteger.cpp

5.15 Tang::AstNodePrint Class Reference

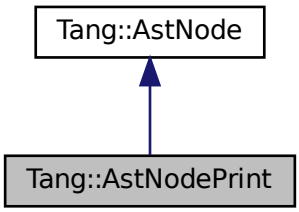
An AstNode that represents a print typeeration.

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

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

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 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.15.4.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.15.4.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.15.4.4 compile()

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

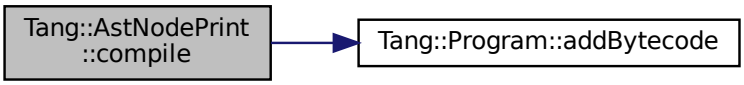
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.15.4.5 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:

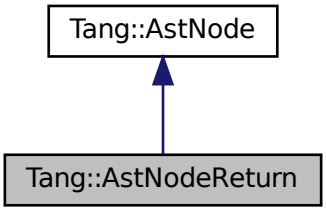
- include/astNodePrint.hpp
- src/astNodePrint.cpp

5.16 Tang::AstNodeReturn Class Reference

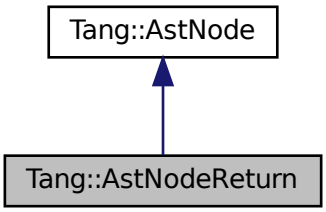
An AstNode that represents a `return` statement.

```
#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 collectIdentifiers (Program &program) const override
  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override
  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const
  Compile a list of the names of functions that are declared in this scope.

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 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.16.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.16.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.16.3.4 compile()

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

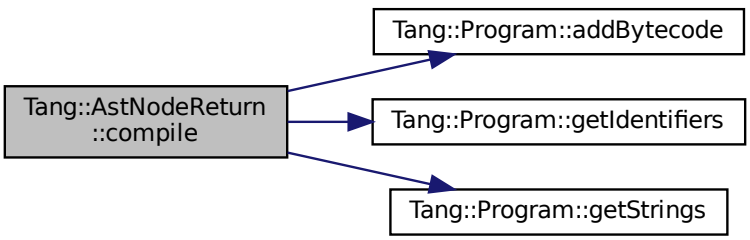
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.16.3.5 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:

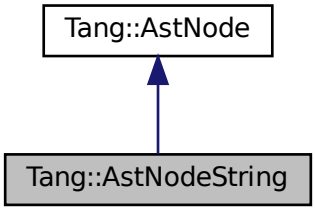
- include/astNodeReturn.hpp
- src/astNodeReturn.cpp

5.17 Tang::AstNodeString Class Reference

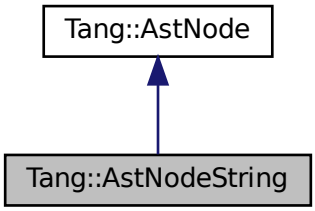
An AstNode that represents a string literal.

```
#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 collectStrings (Program &program) const override
  Compile a list of all string constants in the scope.
- void compileLiteral (Tang::Program &program) const
  Compile the string and push it onto the stack.
- virtual void collectIdentifiers (Program &program) const
  Compile a list of all variables in the scope.
- virtual void collectFunctionDeclarations (Program &program) const
  Compile a list of the names of functions that are declared in this scope.

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 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.17.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

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

5.17.3.3 collectStrings()

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

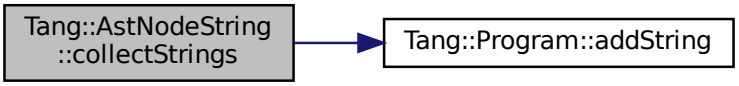
Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.17.3.4 compile()

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

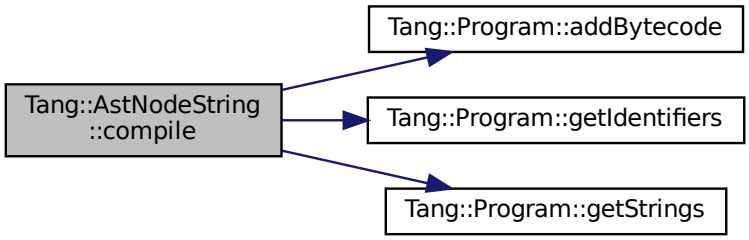
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.17.3.5 compileLiteral()

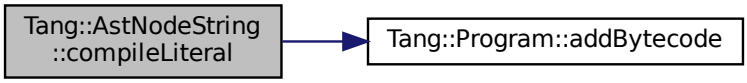
```
void AstNodeString::compileLiteral (
            Tang::Program & program ) const
```

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:



5.17.3.6 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:

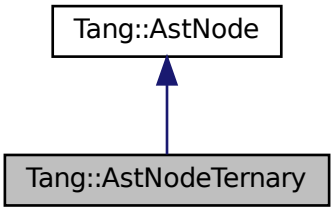
- include/astNodeString.hpp
- src/astNodeString.cpp

5.18 Tang::AstNodeTernary Class Reference

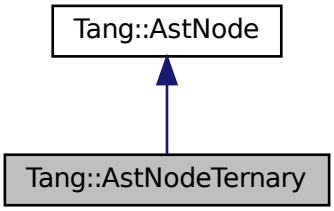
An AstNode that represents a ternary expression.

`#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 collectIdentifiers (Program &program) const override

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

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 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.18.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.18.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.18.3.4 compile()

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

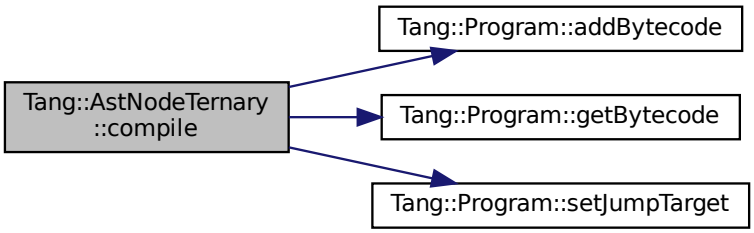
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.18.3.5 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:

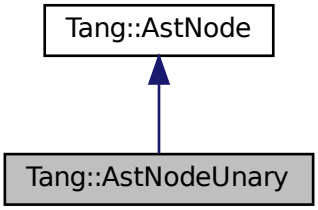
- include/astNodeTernary.hpp
- src/astNodeTernary.cpp

5.19 Tang::AstNodeUnary Class Reference

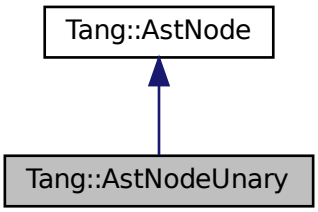
An AstNode that represents a unary negation.

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

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

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 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.19.4.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.19.4.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.19.4.4 compile()

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

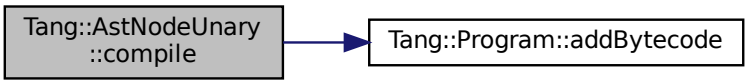
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.19.4.5 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:

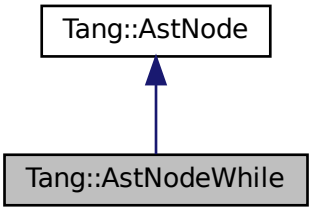
- include/astNodeUnary.hpp
- src/astNodeUnary.cpp

5.20 Tang::AstNodeWhile Class Reference

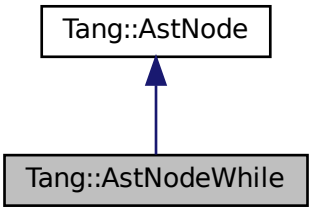
An AstNode that represents a while statement.

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

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

  Compile a list of all string constants in the scope.
- virtual void collectFunctionDeclarations (Program &program) const

  Compile a list of the names of functions that are declared in this scope.

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 collectFunctionDeclarations()

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

Compile a list of the names of functions that are declared in this scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeFunctionDeclaration, and Tang::AstNodeBlock.

5.20.3.2 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.20.3.3 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.20.3.4 compile()

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

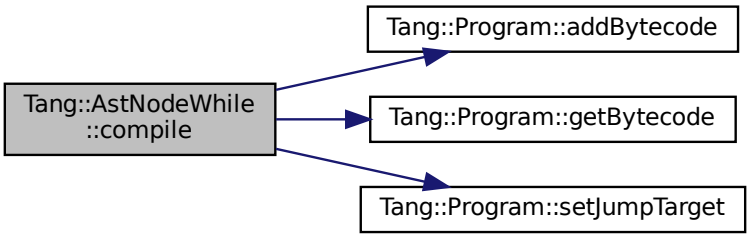
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.20.3.5 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:

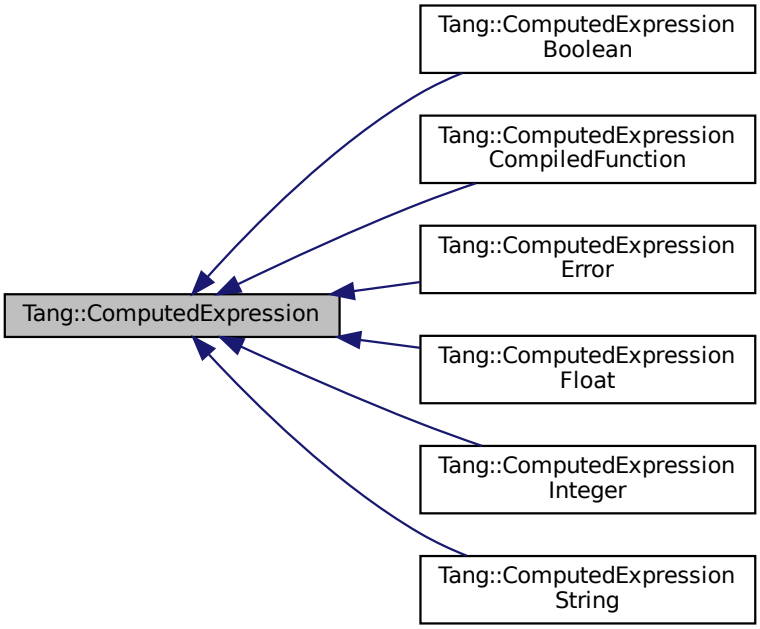
- include/astNodeWhile.hpp
- src/astNodeWhile.cpp

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:

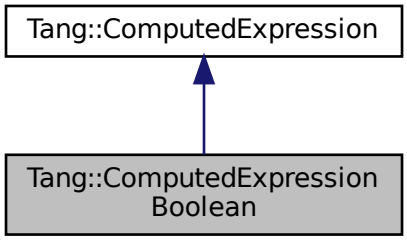
- include/computedExpression.hpp
- src/computedExpression.cpp

5.22 Tang::ComputedExpressionBoolean Class Reference

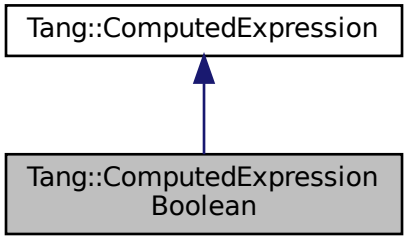
Represents an Boolean that is the result of a computation.

`#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:

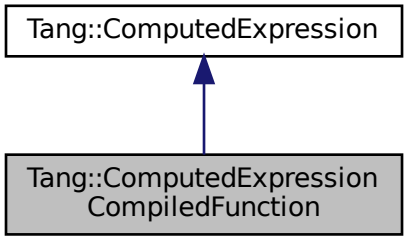
- include/computedExpressionBoolean.hpp
- src/computedExpressionBoolean.cpp

5.23 Tang::ComputedExpressionCompiledFunction Class Reference

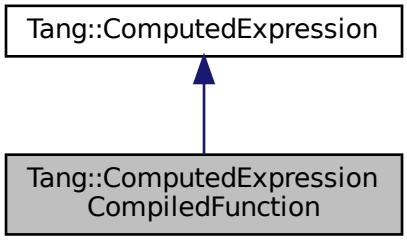
Represents a Compiled Function declared in the script.

```
#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:

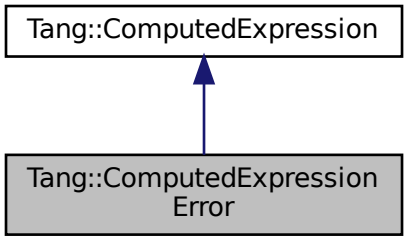
- include/computedExpressionCompiledFunction.hpp
- src/computedExpressionCompiledFunction.cpp

5.24 Tang::ComputedExpressionError Class Reference

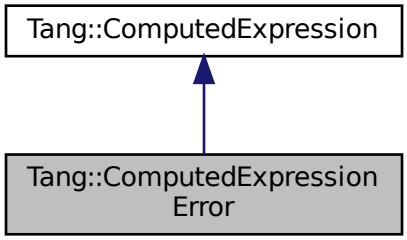
Represents a Runtime Error.

`#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:

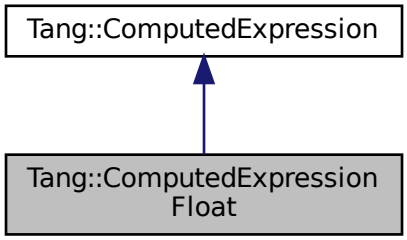
- include/computedExpressionError.hpp
- src/computedExpressionError.cpp

5.25 Tang::ComputedExpressionFloat Class Reference

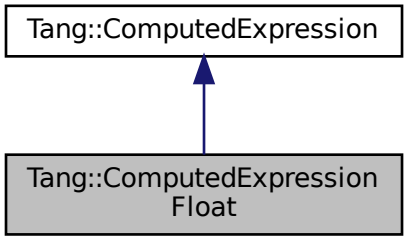
Represents a Float that is the result of a computation.

`#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]
```

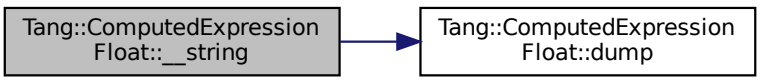
Perform a type cast to string.

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:

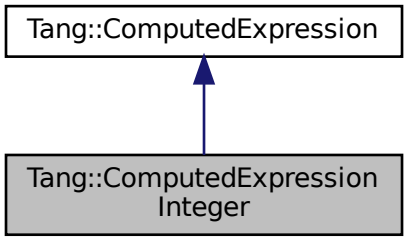
- include/computedExpressionFloat.hpp
- src/computedExpressionFloat.cpp

5.26 Tang::ComputedExpressionInteger Class Reference

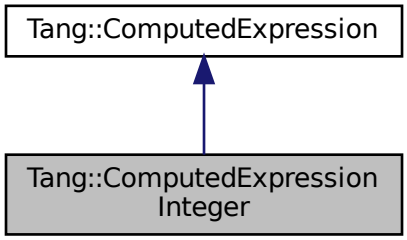
Represents an Integer that is the result of a computation.

`#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]
```

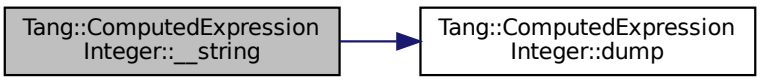
Perform a type cast to string.

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:

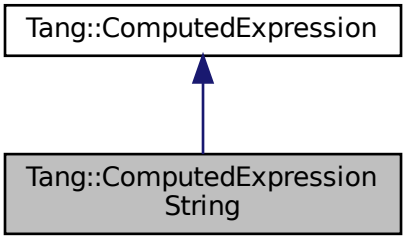
- include/computedExpressionInteger.hpp
- src/computedExpressionInteger.cpp

5.27 Tang::ComputedExpressionString Class Reference

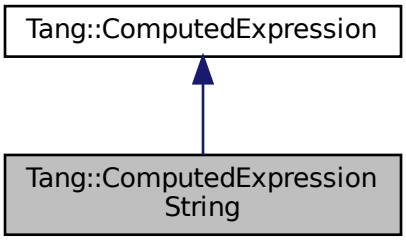
Represents a String that is the result of a computation.

```
#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:

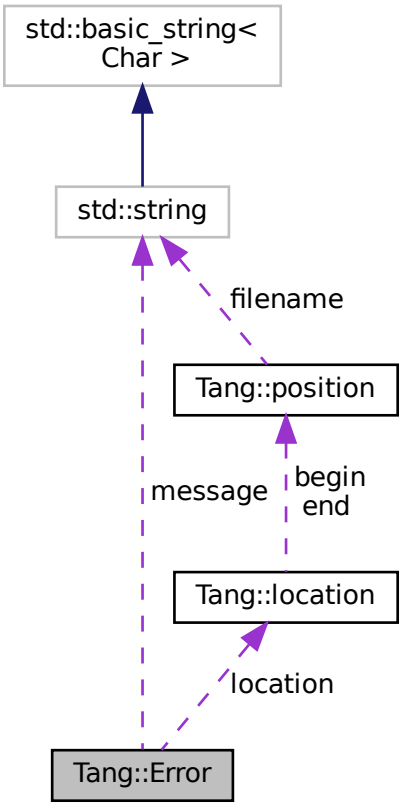
- include/computedExpressionString.hpp
- src/computedExpressionString.cpp

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:

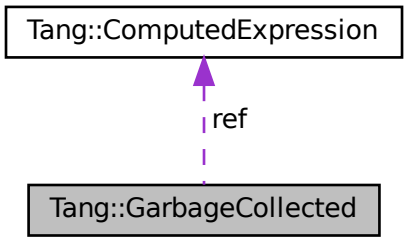
- include/error.hpp
- src/error.cpp

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.

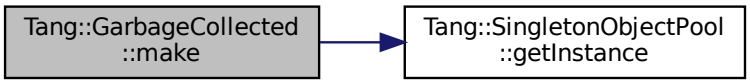
Parameters

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"!()

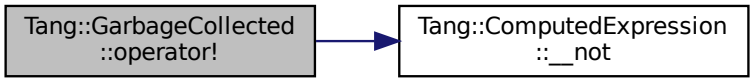
```
GarbageCollected GarbageCollected::operator!  ( ) const
```

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.

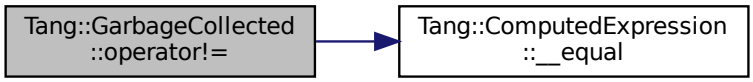
Parameters

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.

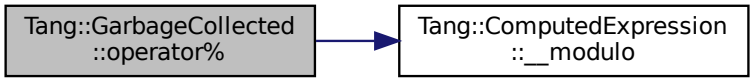
Parameters

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.

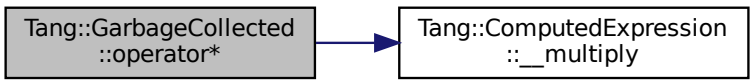
Parameters

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.

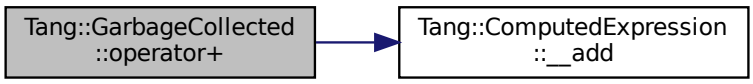
Parameters

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]

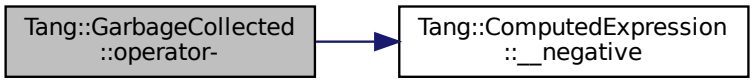
```
GarbageCollected GarbageCollected::operator- ( ) const
```

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.

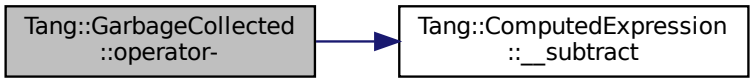
Parameters

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.

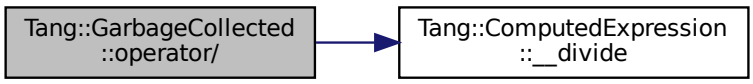
Parameters

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.

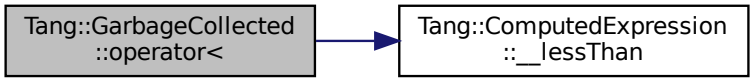
Parameters

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]

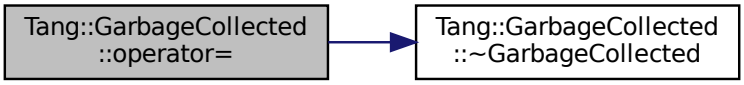
```
GarbageCollected& Tang::GarbageCollected::operator= (
            const GarbageCollected & other )  [inline]
```

Copy Assignment.

Parameters

The	other GarbageCollected object.

Here is the call graph for this function:



5.29.3.15 operator=() [2/2]

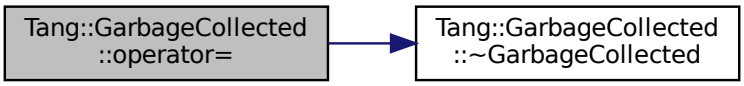
```
GarbageCollected& Tang::GarbageCollected::operator= (
            GarbageCollected && other )  [inline]
```

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.

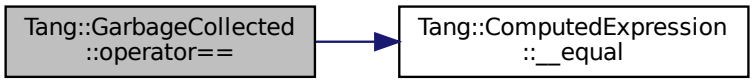
Parameters

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.

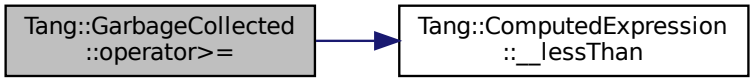
Parameters

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:

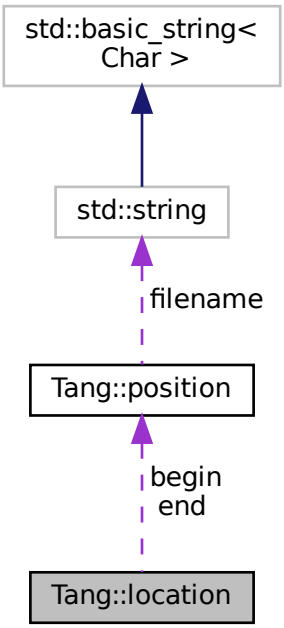
- include/garbageCollected.hpp
- src/garbageCollected.cpp

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:

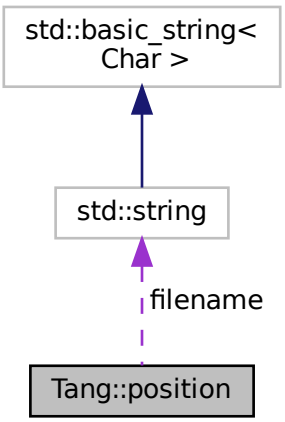
- build/generated/location.hh

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:

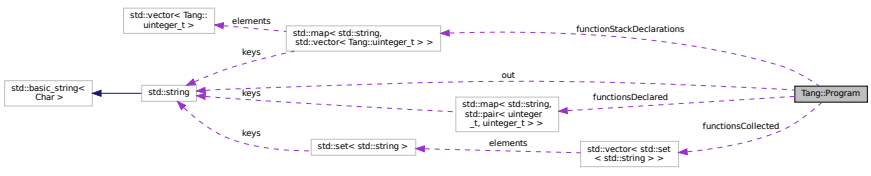
- build/generated/location.hh

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()

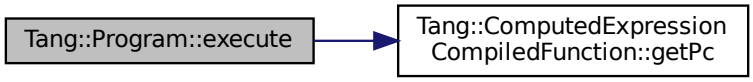
```
Program & Program::execute ( )
```

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:

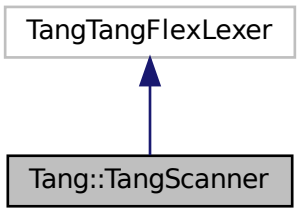
- include/tangBase.hpp
- src/tangBase.cpp

5.35 Tang::TangScanner Class Reference

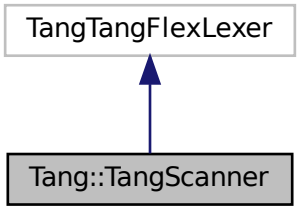
The Flex lexer class for the main Tang language.

`#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

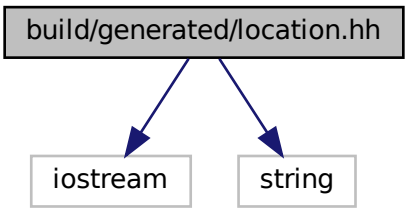
File Documentation

6.1 build/generated/location.hh File Reference

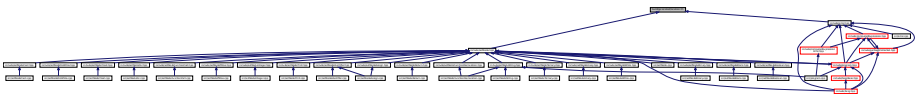
Define the Tang ::location class.

```
#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

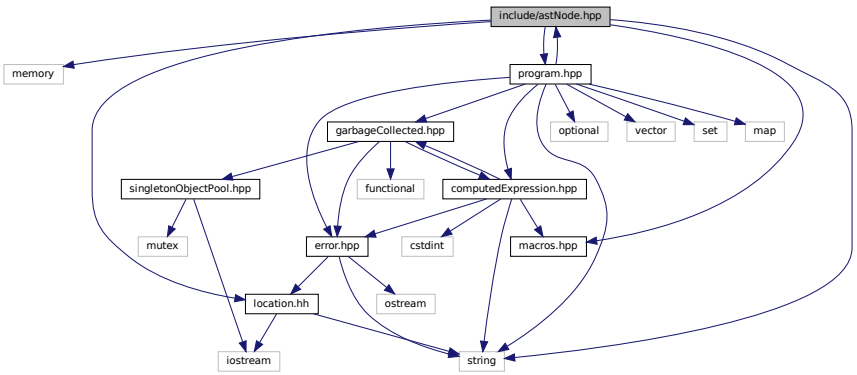
ostr	the destination output stream
pos	a reference to the position to redirect

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).

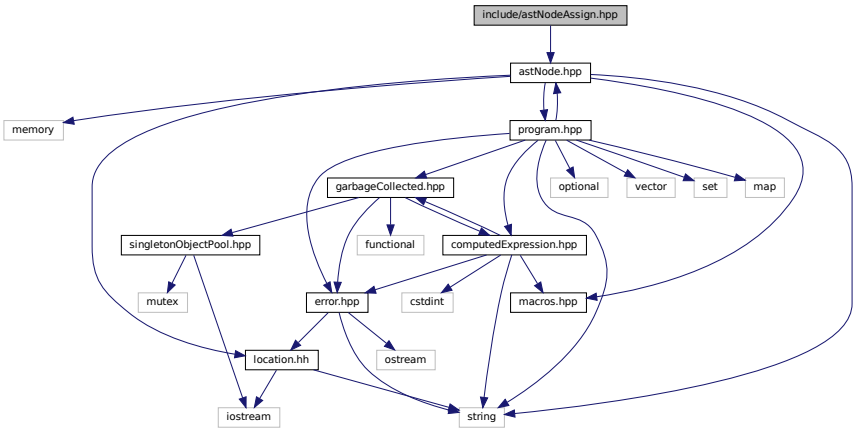
6.2.1 Detailed Description

Declare the Tang::AstNode base class.

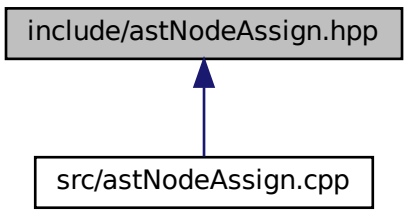
6.3 include/astNodeAssign.hpp File Reference

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

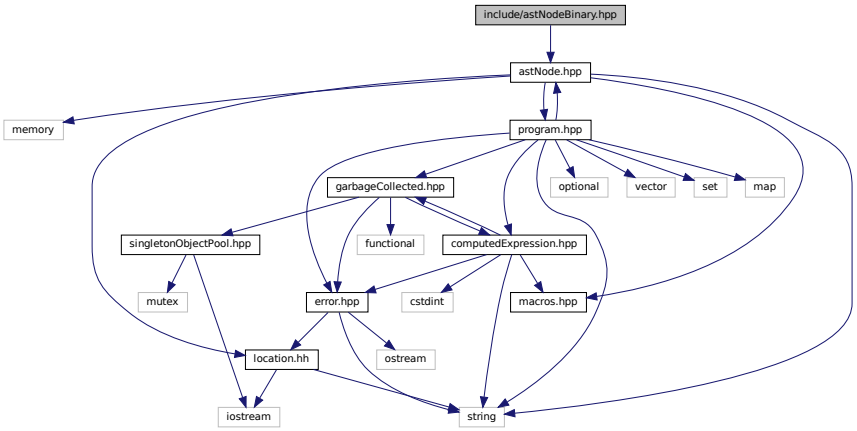
Declare the Tang::AstNodeAssign class.

6.4 include/astNodeBinary.hpp File Reference

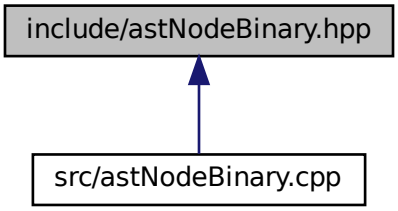
Declare the Tang::AstNodeBinary class.

```
#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

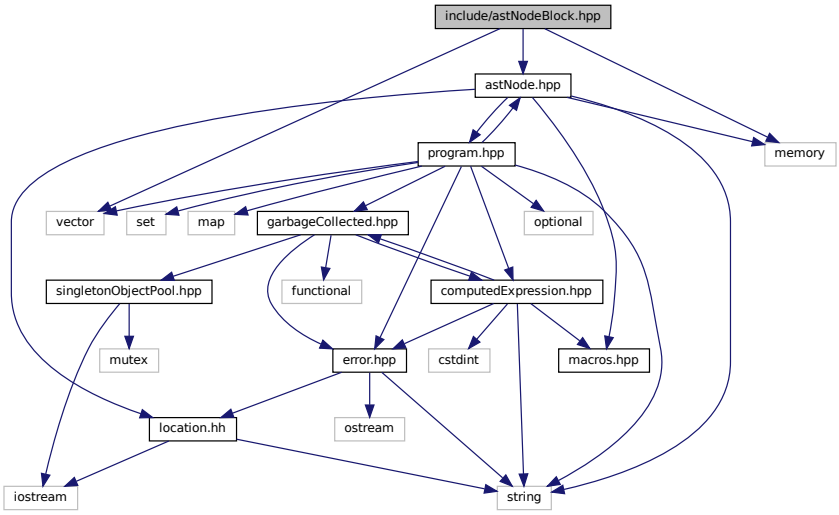
Declare the Tang::AstNodeBinary class.

6.5 include/astNodeBlock.hpp File Reference

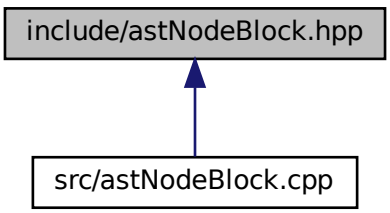
Declare the Tang::AstNodeBlock class.

```
#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

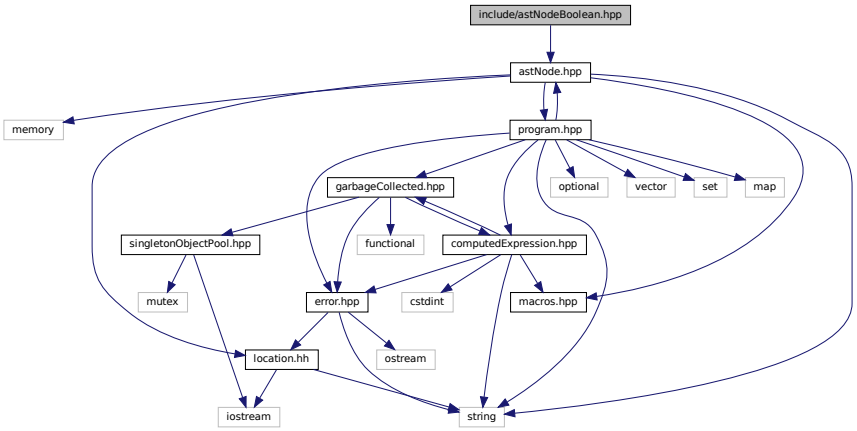
Declare the Tang::AstNodeBlock class.

6.6 include/astNodeBoolean.hpp File Reference

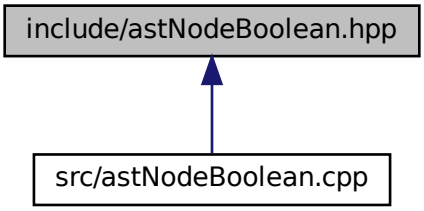
Declare the Tang::AstNodeBoolean class.

```
#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

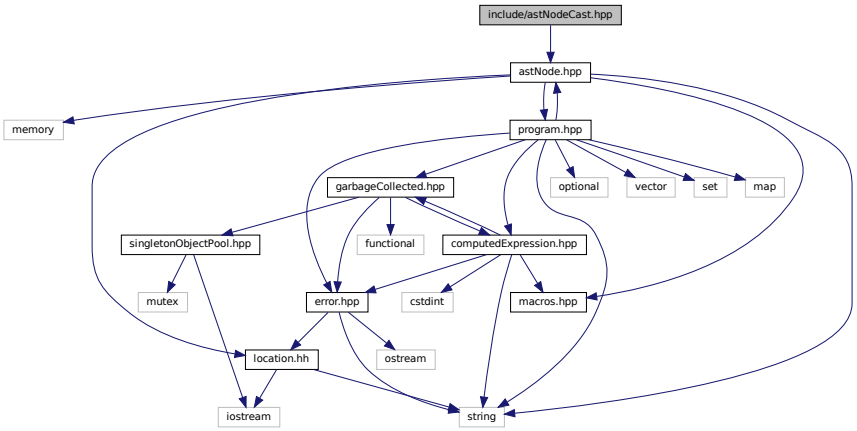
Declare the Tang::AstNodeBoolean class.

6.7 include/astNodeCast.hpp File Reference

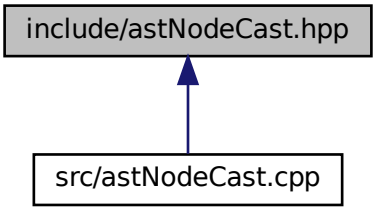
Declare the Tang::AstNodeCast class.

```
#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

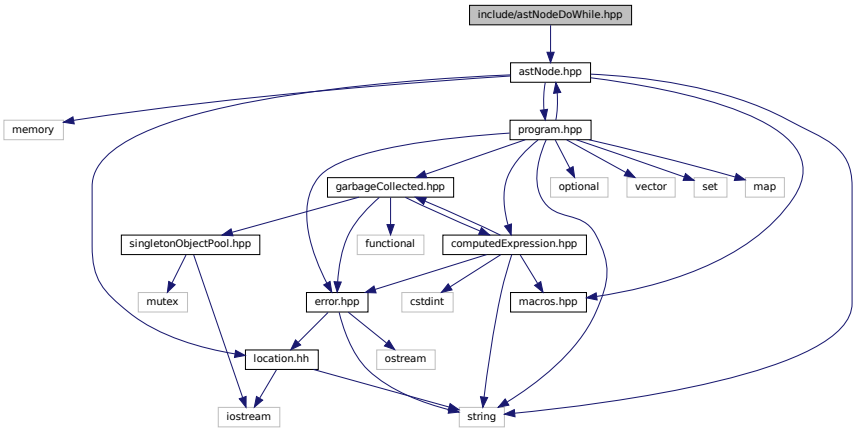
Declare the Tang::AstNodeCast class.

6.8 include/astNodeDoWhile.hpp File Reference

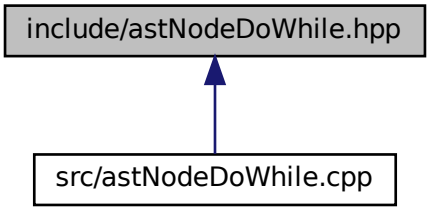
Declare the Tang::AstNodeDoWhile class.

```
#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

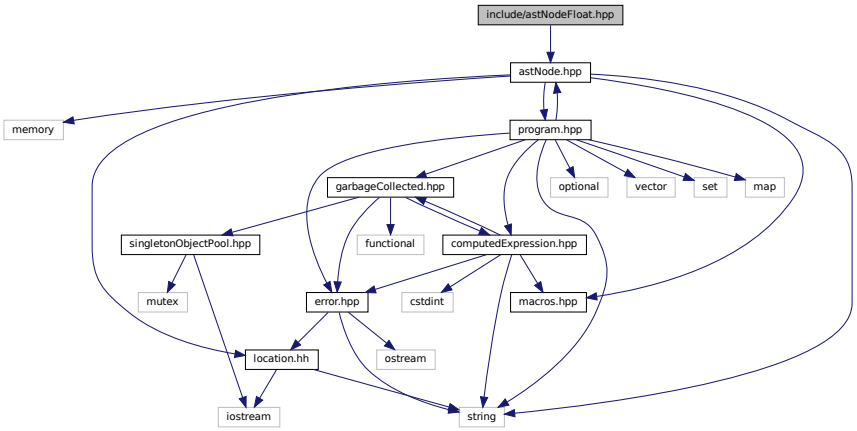
Declare the Tang::AstNodeDoWhile class.

6.9 include/astNodeFloat.hpp File Reference

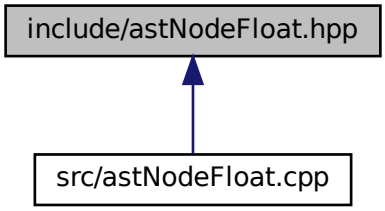
Declare the Tang::AstNodeFloat class.

```
#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

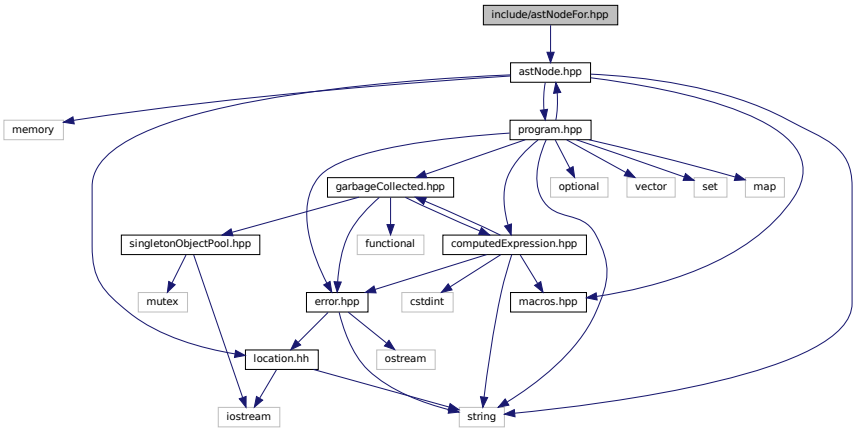
Declare the Tang::AstNodeFloat class.

6.10 include/astNodeFor.hpp File Reference

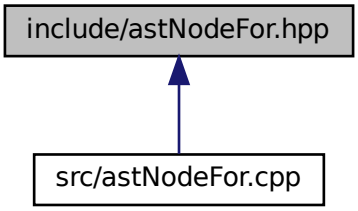
Declare the Tang::AstNodeFor class.

```
#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

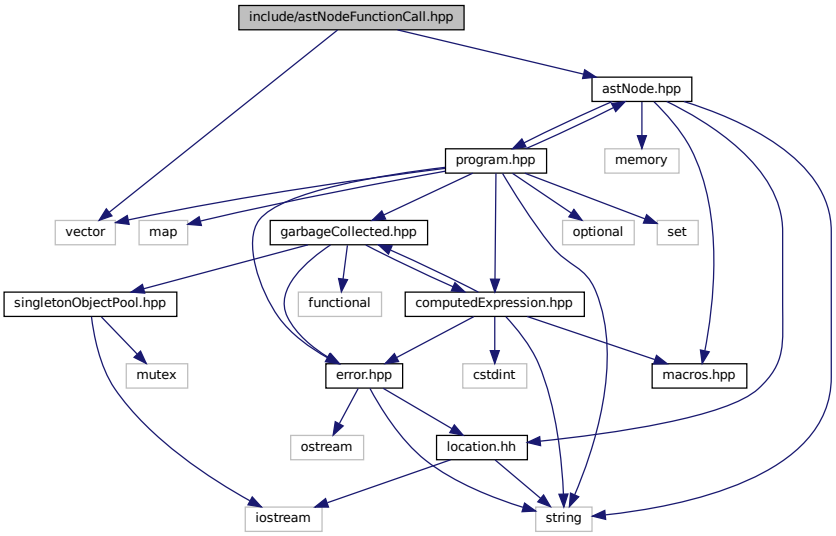
Declare the Tang::AstNodeFor class.

6.11 include/astNodeFunctionCall.hpp File Reference

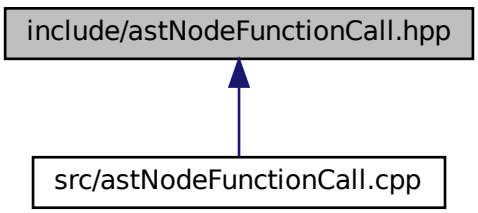
Declare the Tang::AstNodeFunctionCall class.

```
#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

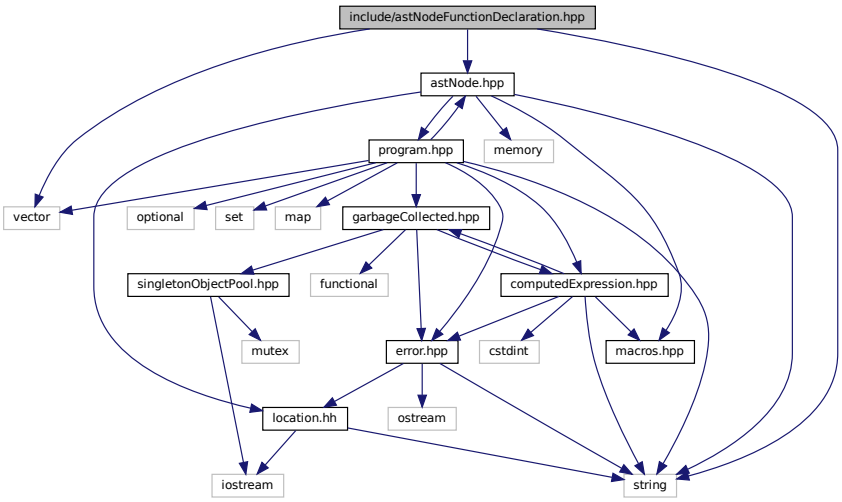
Declare the Tang::AstNodeFunctionCall class.

6.12 include/astNodeFunctionDeclaration.hpp File Reference

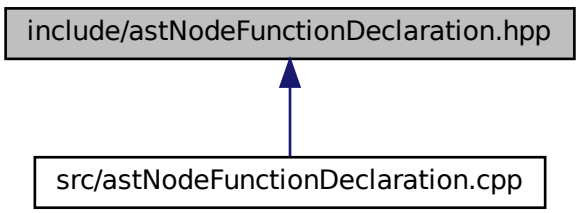
Declare the Tang::AstNodeFunctionDeclaration class.

```
#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

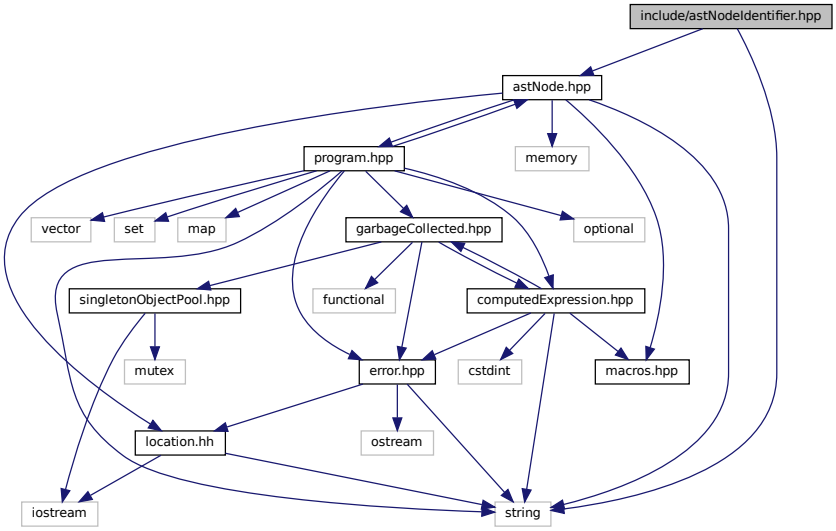
Declare the Tang::AstNodeFunctionDeclaration class.

6.13 include/astNodeIdentifier.hpp File Reference

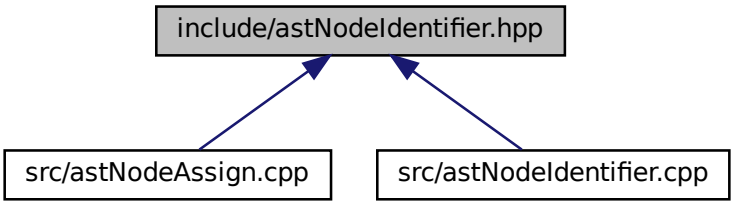
Declare the Tang::AstNodeIdentifier class.

```
#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

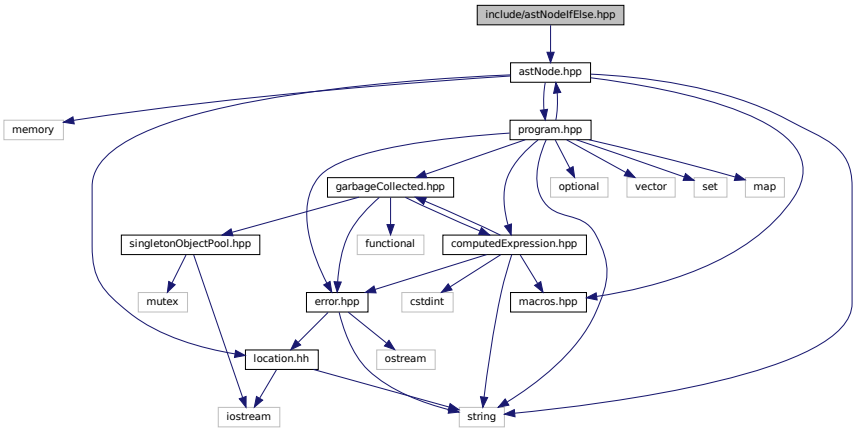
Declare the Tang::AstNodeIdentifier class.

6.14 include/astNodeIfElse.hpp File Reference

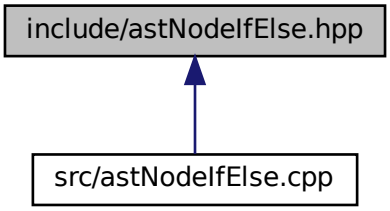
Declare the Tang::AstNodeIfElse class.

```
#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.

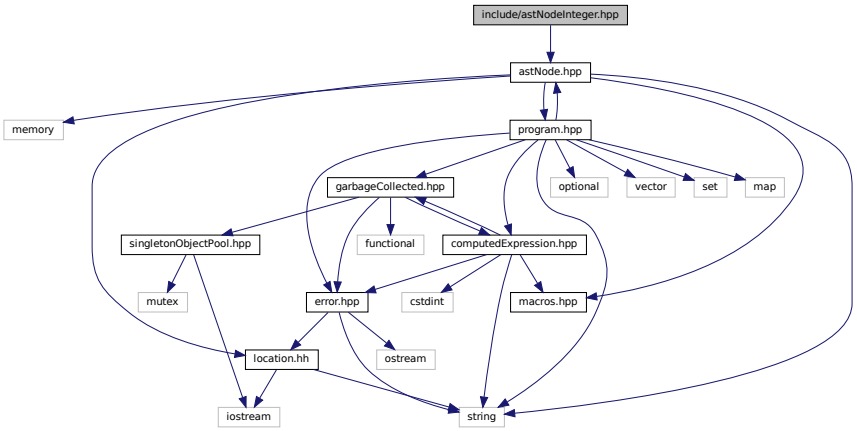
6.14.1 Detailed Description

Declare the Tang::AstNodeIfElse class.

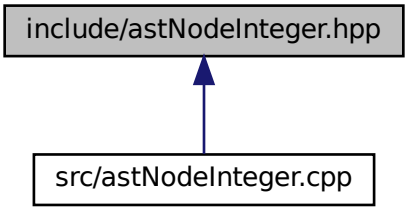
6.15 include/astNodeInteger.hpp File Reference

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

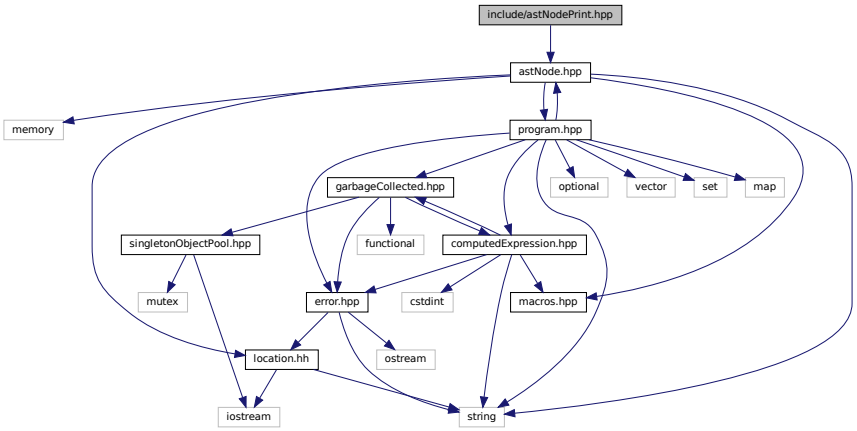
Declare the Tang::AstNodeInteger class.

6.16 include/astNodePrint.hpp File Reference

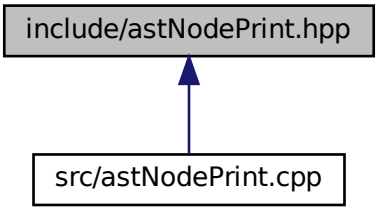
Declare the Tang::AstNodePrint class.

```
#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

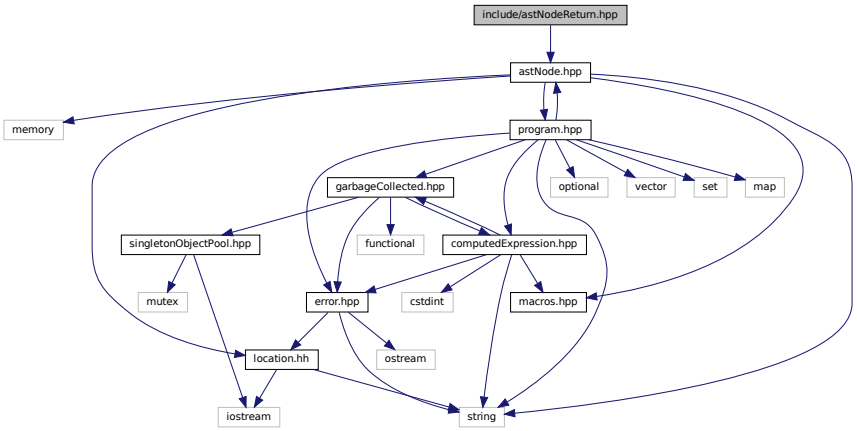
Declare the Tang::AstNodePrint class.

6.17 include/astNodeReturn.hpp File Reference

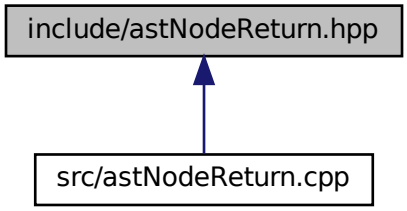
Declare the Tang::AstNodeReturn class.

```
#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

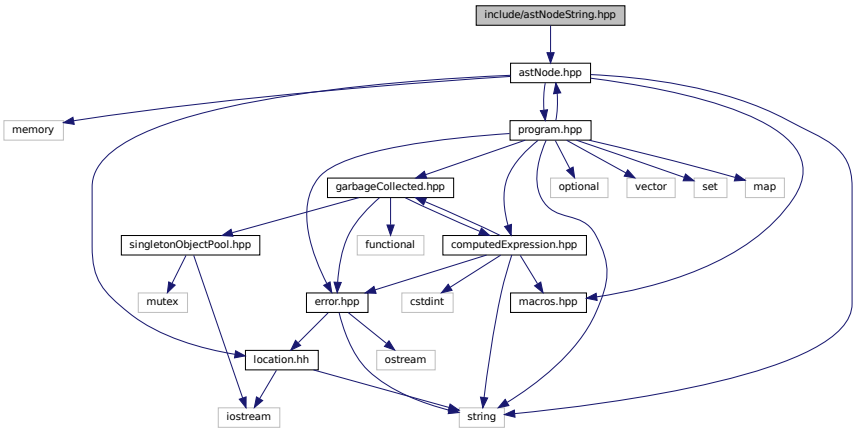
Declare the Tang::AstNodeReturn class.

6.18 include/astNodeString.hpp File Reference

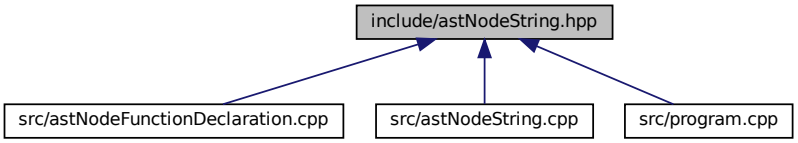
Declare the Tang::AstNodeString class.

```
#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.

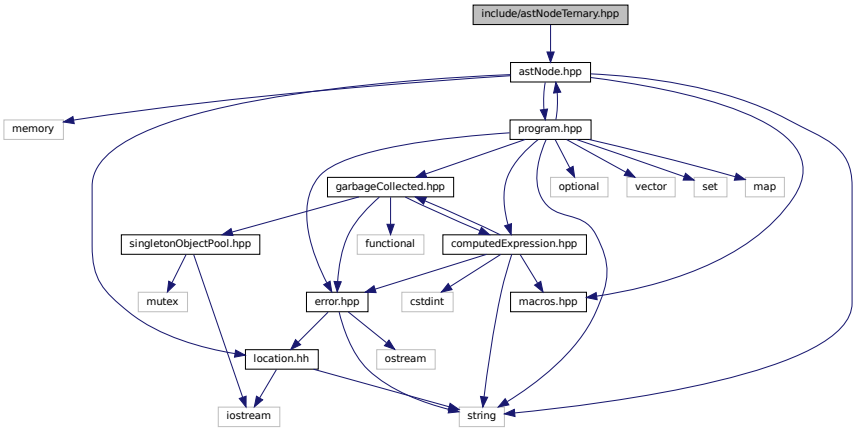
6.18.1 Detailed Description

Declare the Tang::AstNodeString class.

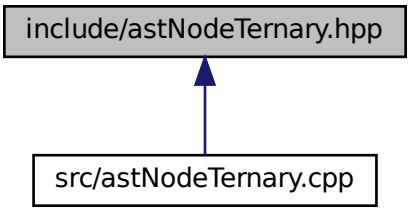
6.19 include/astNodeTernary.hpp File Reference

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

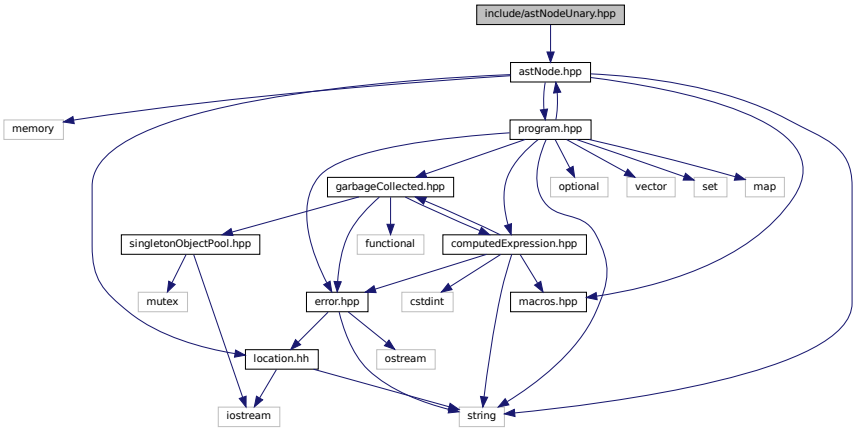
Declare the Tang::AstNodeTernary class.

6.20 include/astNodeUnary.hpp File Reference

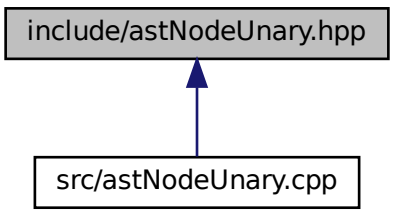
Declare the Tang::AstNodeUnary class.

```
#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.

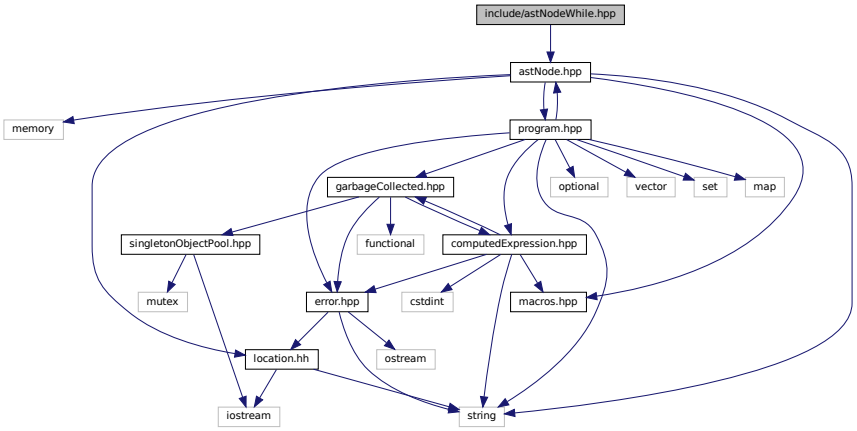
6.20.1 Detailed Description

Declare the Tang::AstNodeUnary class.

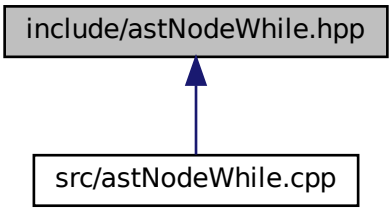
6.21 include/astNodeWhile.hpp File Reference

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

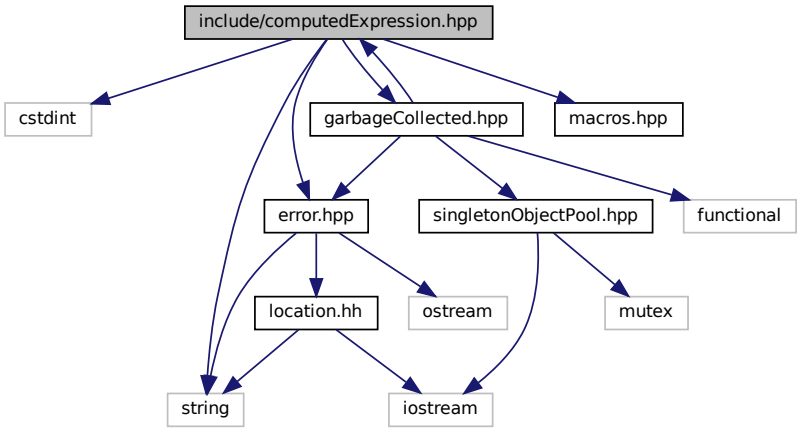
Declare the Tang::AstNodeWhile class.

6.22 include/computedExpression.hpp File Reference

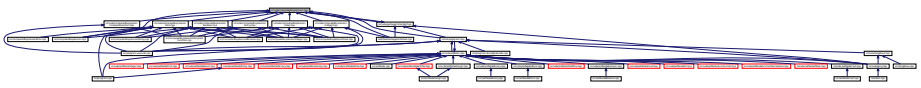
Declare the Tang::ComputedExpression base class.

```
#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

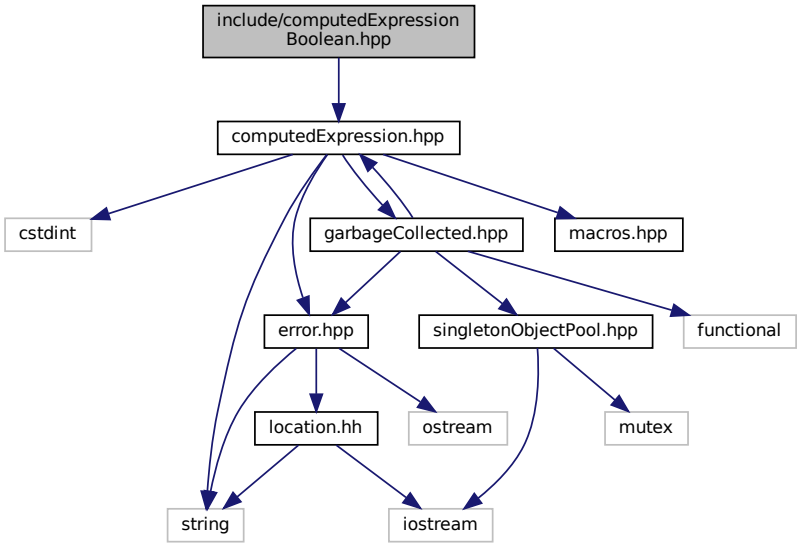
Declare the Tang::ComputedExpression base class.

6.23 include/computedExpressionBoolean.hpp File Reference

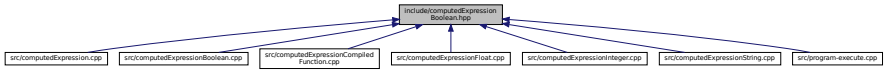
Declare the Tang::ComputedExpressionBoolean class.

```
#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

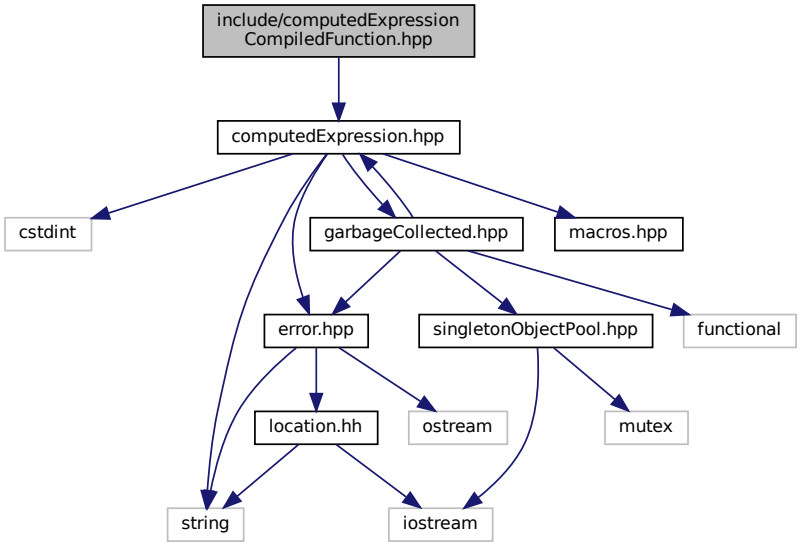
Declare the Tang::ComputedExpressionBoolean class.

6.24 include/computedExpressionCompiledFunction.hpp File Reference

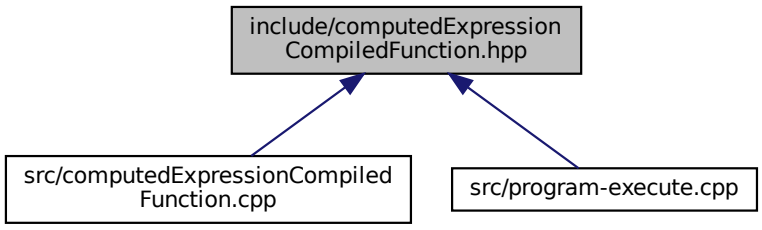
Declare the Tang::ComputedExpressionCompiledFunction class.

```
#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

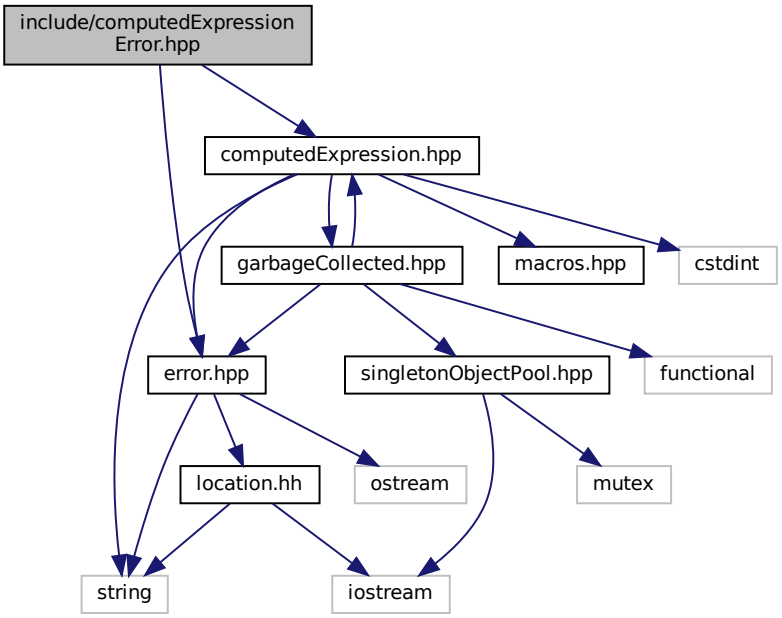
Declare the Tang::ComputedExpressionCompiledFunction class.

6.25 include/computedExpressionError.hpp File Reference

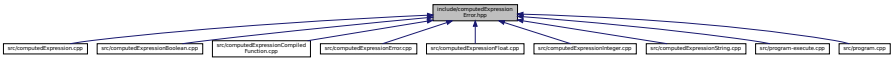
Declare the Tang::ComputedExpressionError class.

```
#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

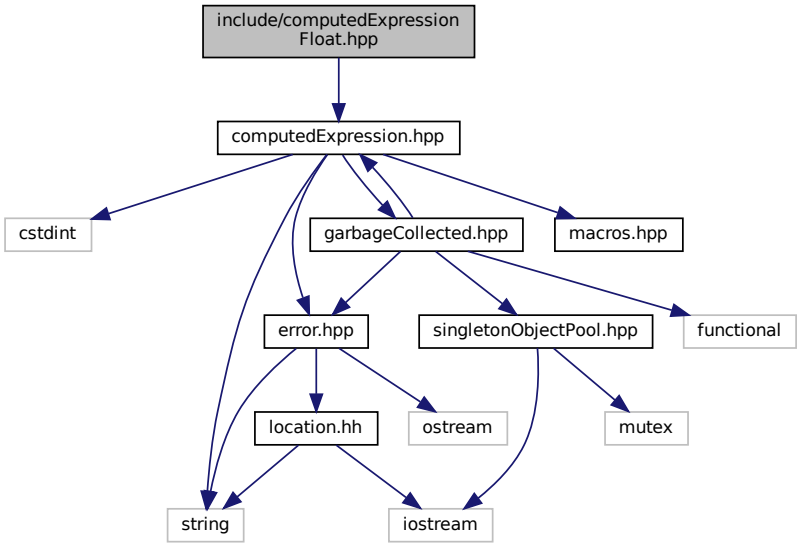
Declare the Tang::ComputedExpressionError class.

6.26 include/computedExpressionFloat.hpp File Reference

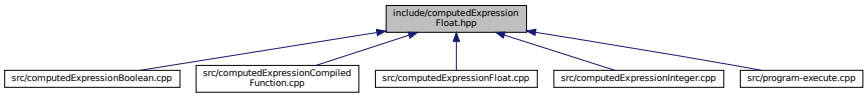
Declare the Tang::ComputedExpressionFloat class.

```
#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

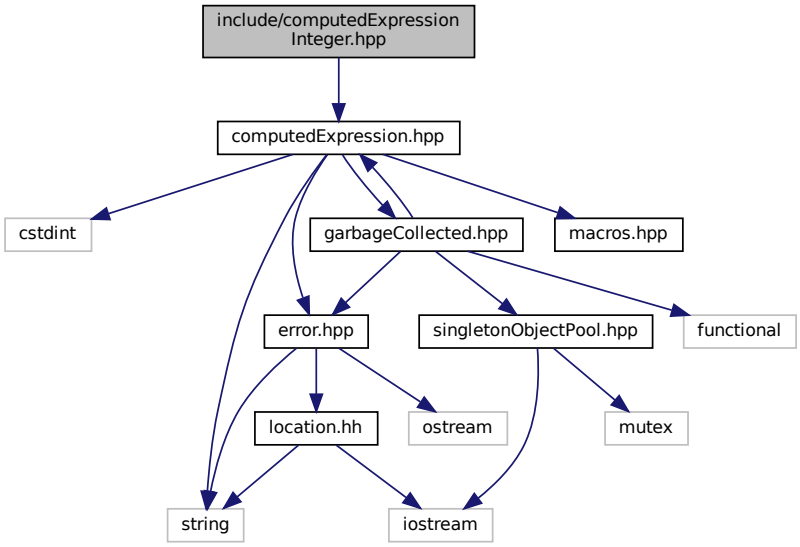
Declare the Tang::ComputedExpressionFloat class.

6.27 include/computedExpressionInteger.hpp File Reference

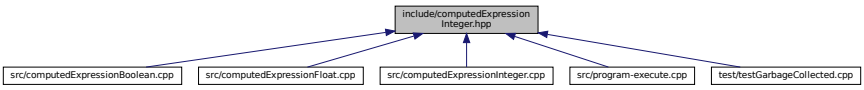
Declare the Tang::ComputedExpressionInteger class.

```
#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

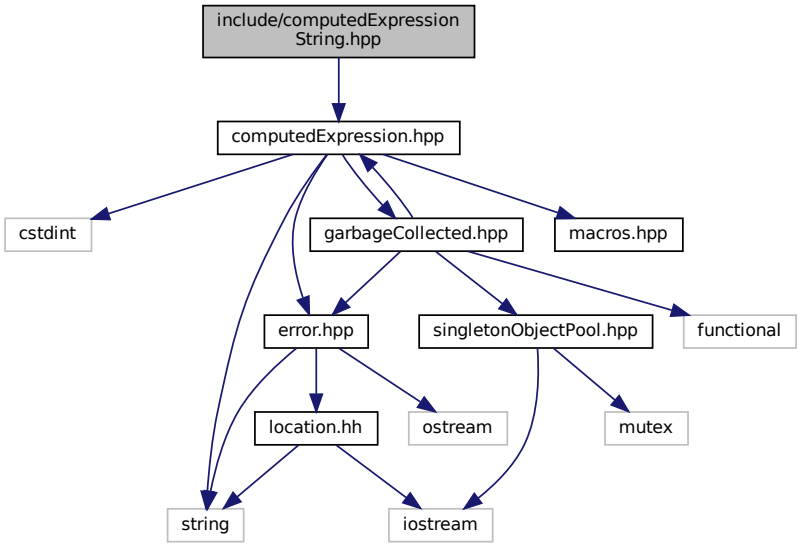
Declare the Tang::ComputedExpressionInteger class.

6.28 include/computedExpressionString.hpp File Reference

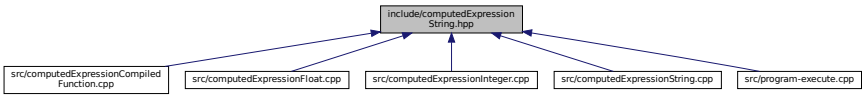
Declare the Tang::ComputedExpressionString class.

```
#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

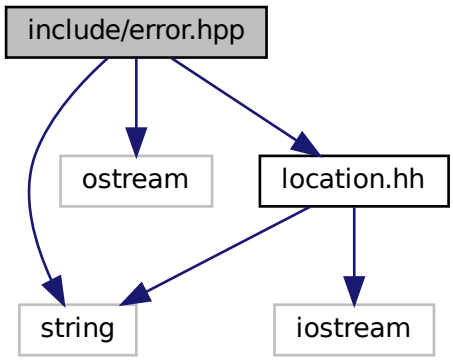
Declare the Tang::ComputedExpressionString class.

6.29 include/error.hpp File Reference

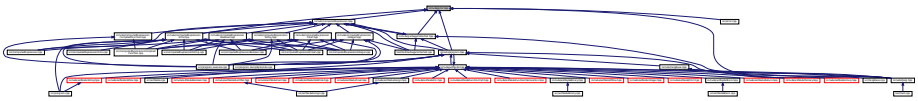
Declare the Tang::Error class used to describe syntax and runtime errors.

```
#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

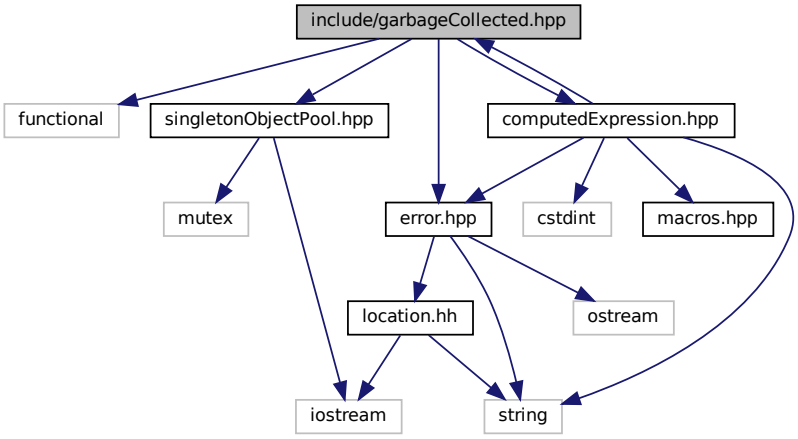
Declare the Tang::Error class used to describe syntax and runtime errors.

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.

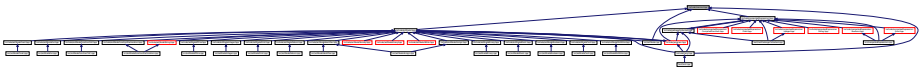
6.30.1 Detailed Description

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.

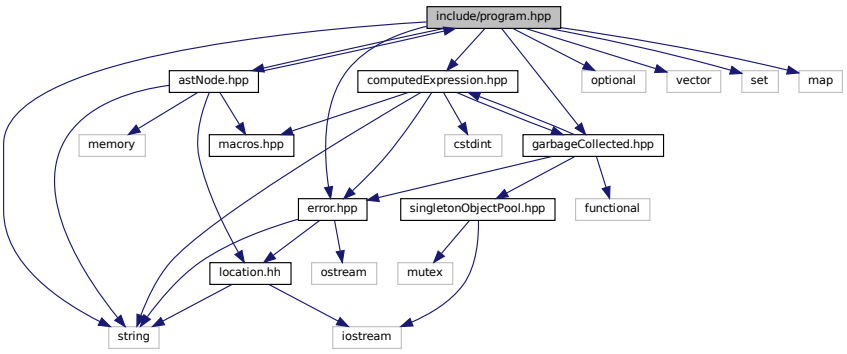
6.33 include/program.hpp File Reference

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

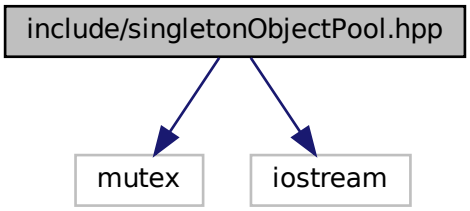
Declare the Tang::Program class used to compile and execute source code.

6.34 include/singletonObjectPool.hpp File Reference

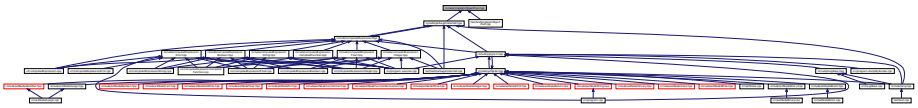
Declare the Tang::SingletonObjectPool class.

```
#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.

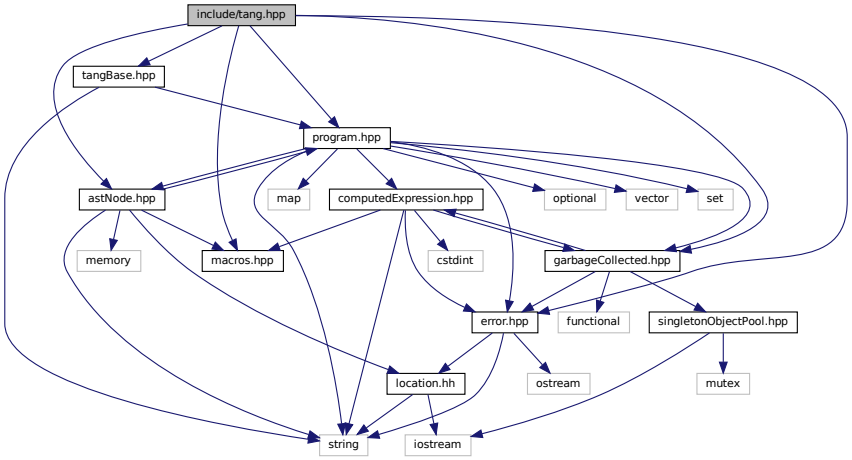
6.35 include/tang.hpp File Reference

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

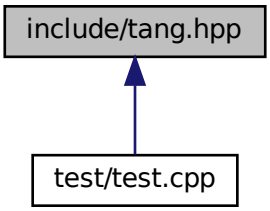
```
#include "macros.hpp"
#include "tangBase.hpp"
#include "astNode.hpp"
#include "error.hpp"
#include "garbageCollected.hpp"
```

```
#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

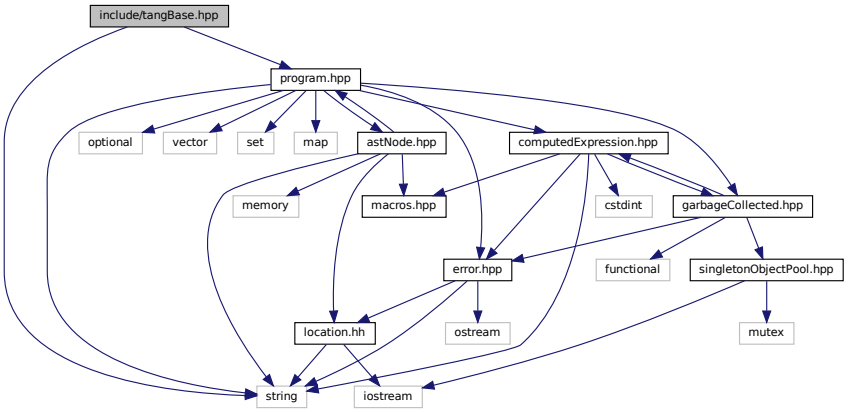
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

6.36 include/tangBase.hpp File Reference

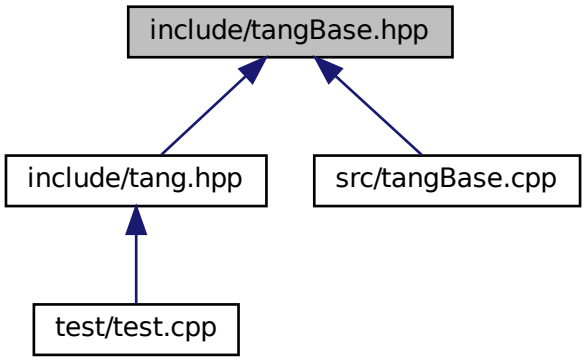
Declare the Tang::TangBase class used to interact with Tang.

```
#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

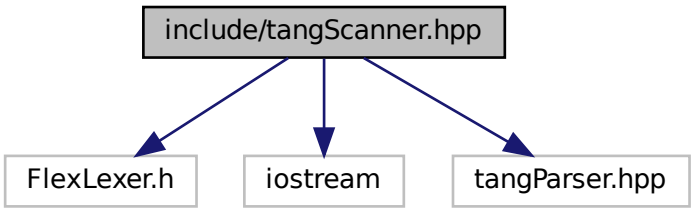
Declare the Tang::TangBase class used to interact with Tang.

6.37 include/tangScanner.hpp File Reference

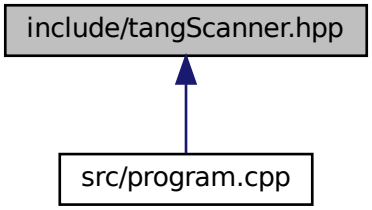
Declare the Tang::TangScanner used to tokenize a Tang script.

```
#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

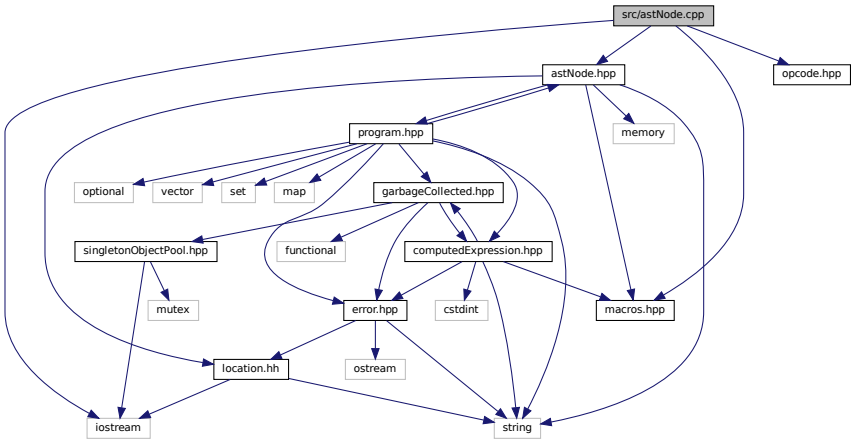
Declare the Tang::TangScanner used to tokenize a Tang script.

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

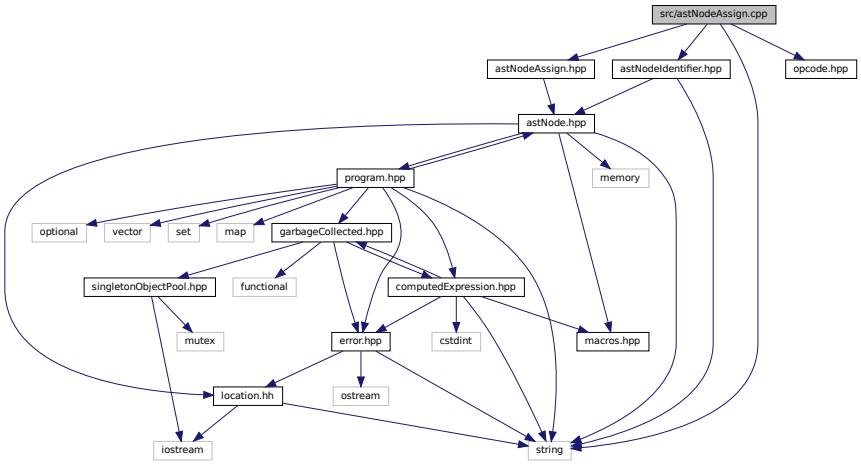
Define the Tang::AstNode class.

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:



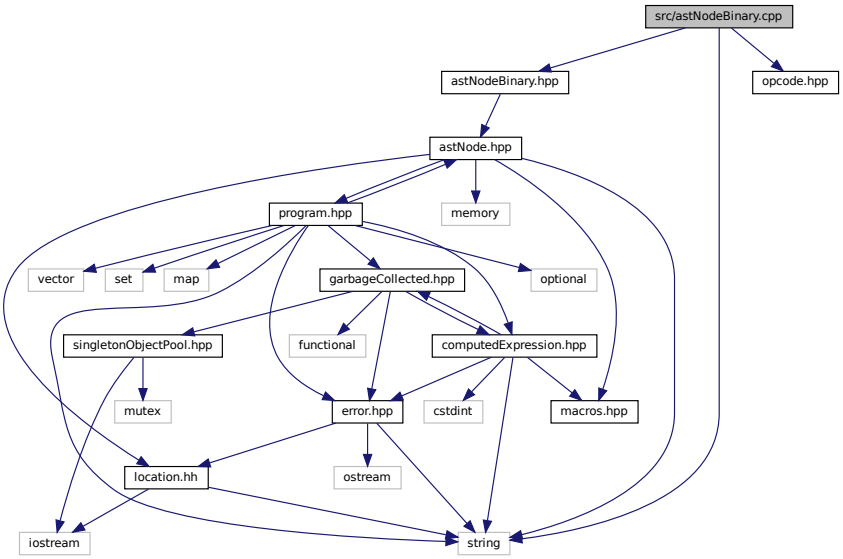
6.39.1 Detailed Description

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

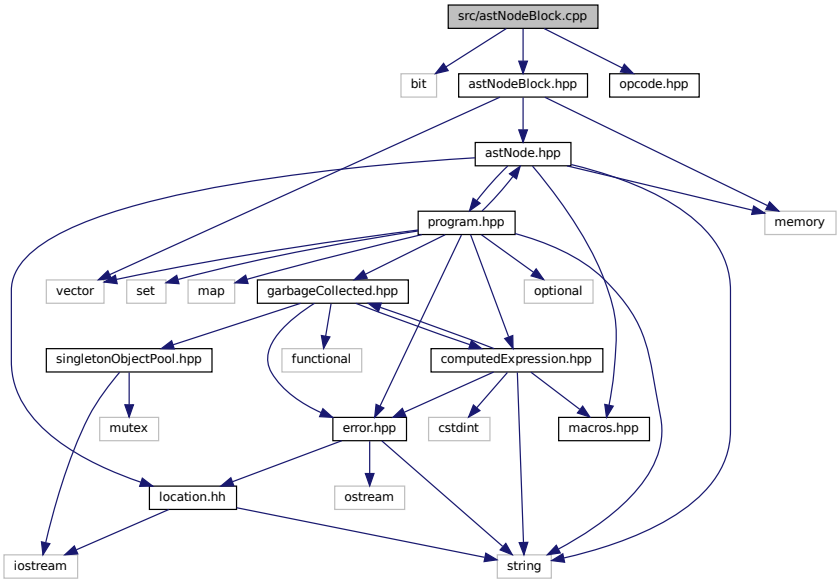
Define the Tang::AstNodeBinary class.

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

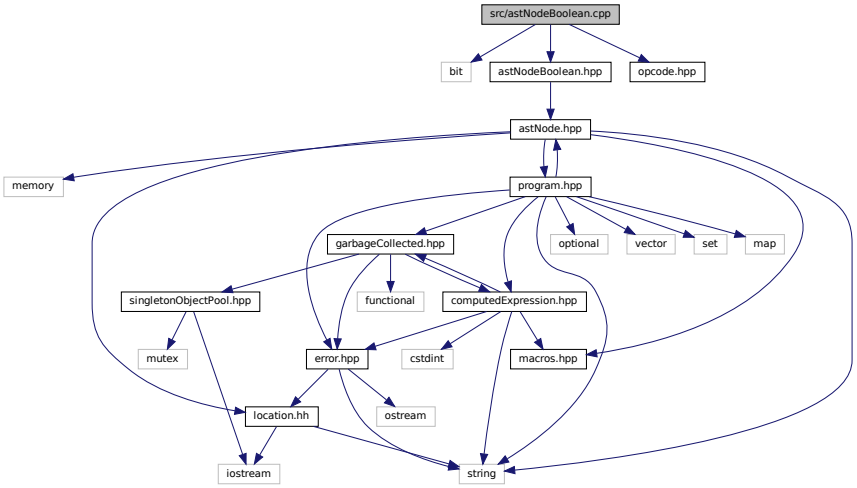
Define the Tang::AstNodeBlock class.

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:



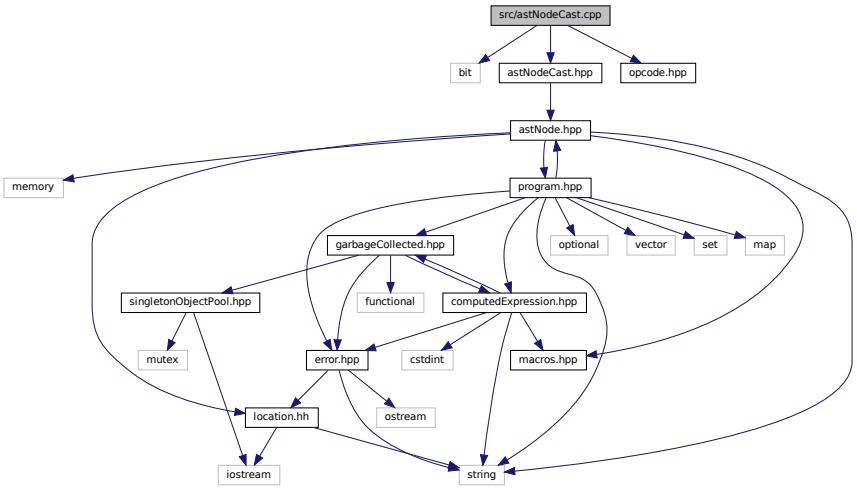
6.42.1 Detailed Description

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

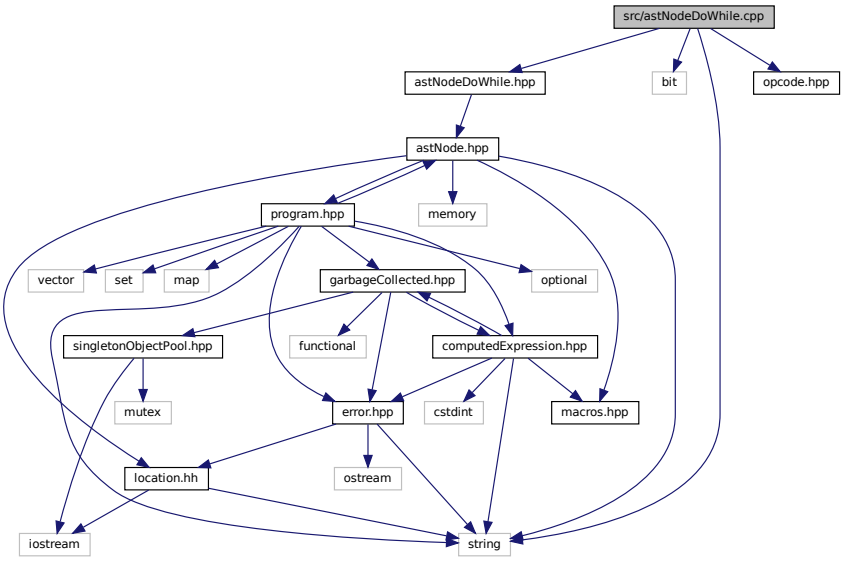
Define the Tang::AstNodeCast class.

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.

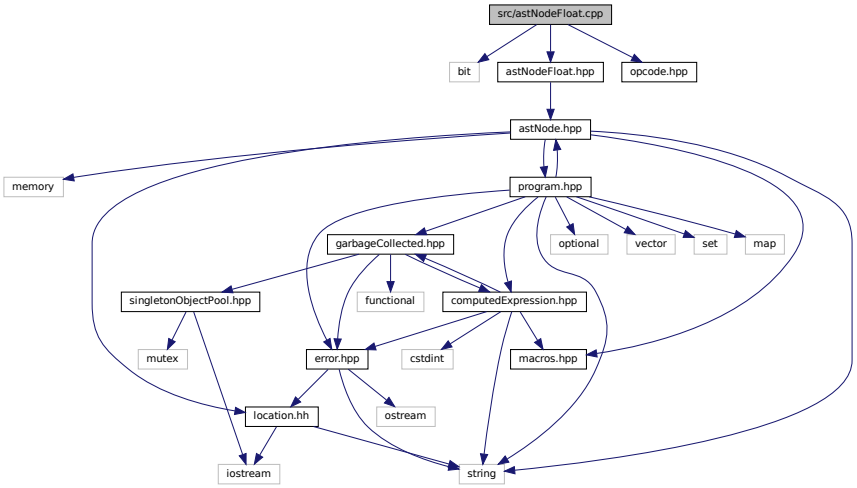
6.45 src/astNodeFloat.cpp File Reference

Define the Tang::AstNodeFloat class.

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

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

Include dependency graph for astNodeFloat.cpp:



6.45.1 Detailed Description

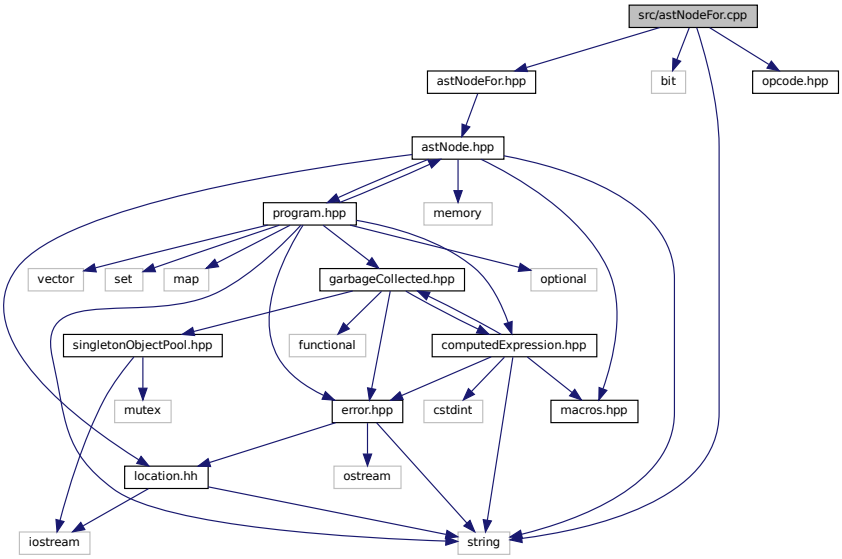
Define the Tang::AstNodeFloat class.

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

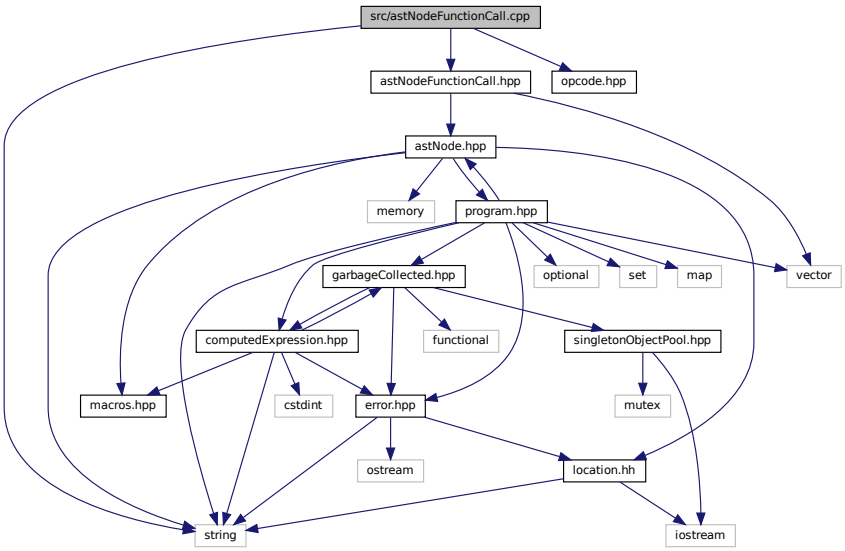
Define the Tang::AstNodeFor class.

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

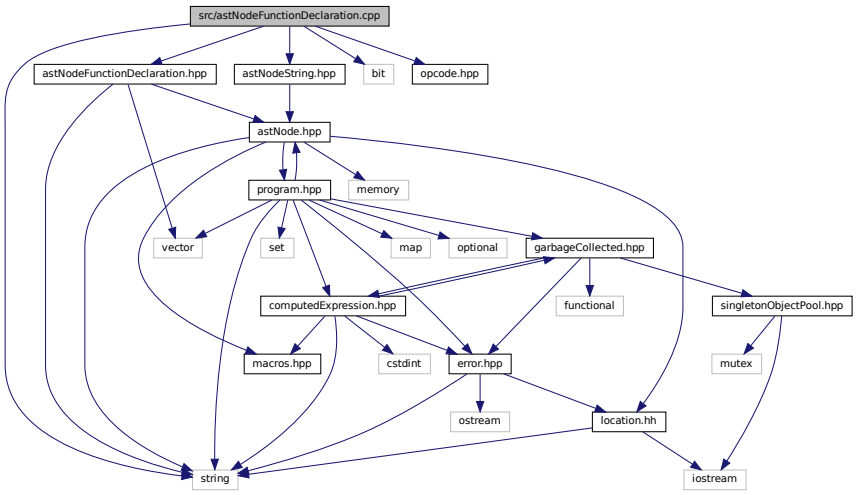
Define the Tang::AstNodeFunctionCall class.

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

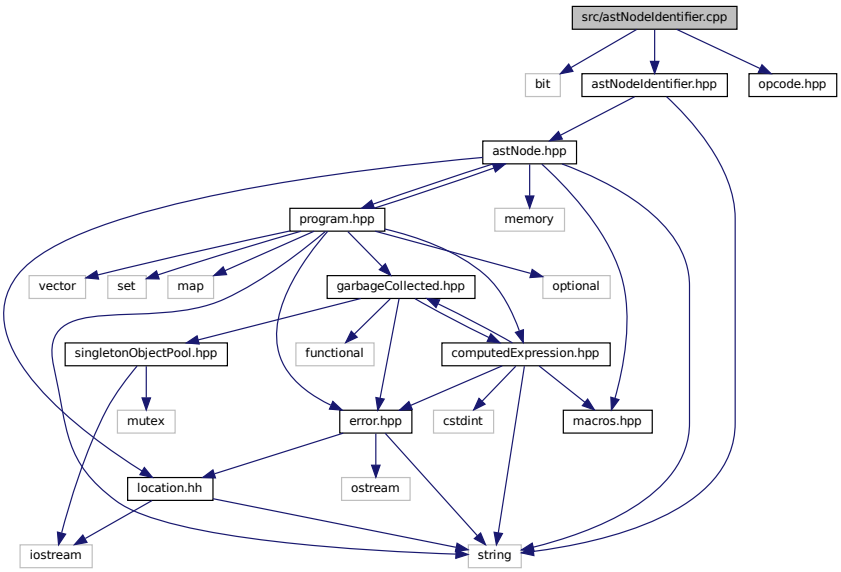
Define the Tang::AstNodeFunctionDeclaration class.

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

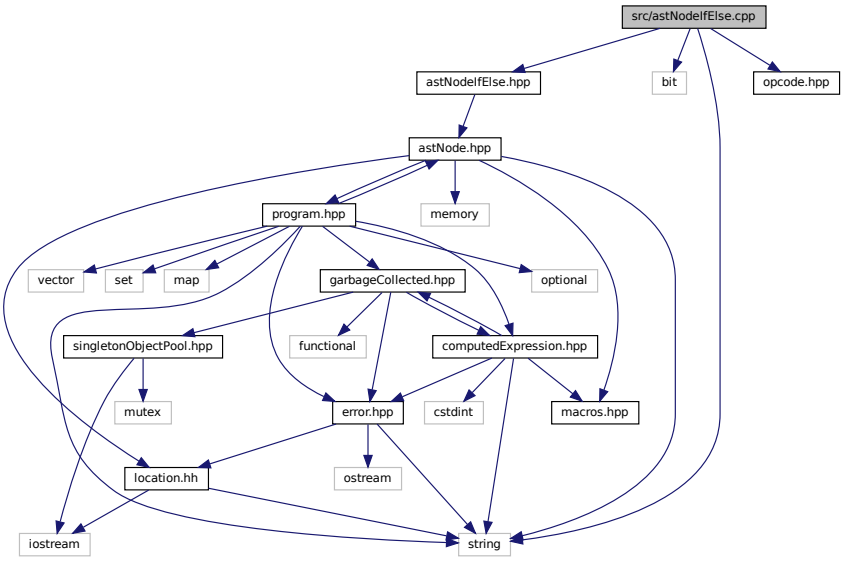
Define the Tang::AstNodeIdentifier class.

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

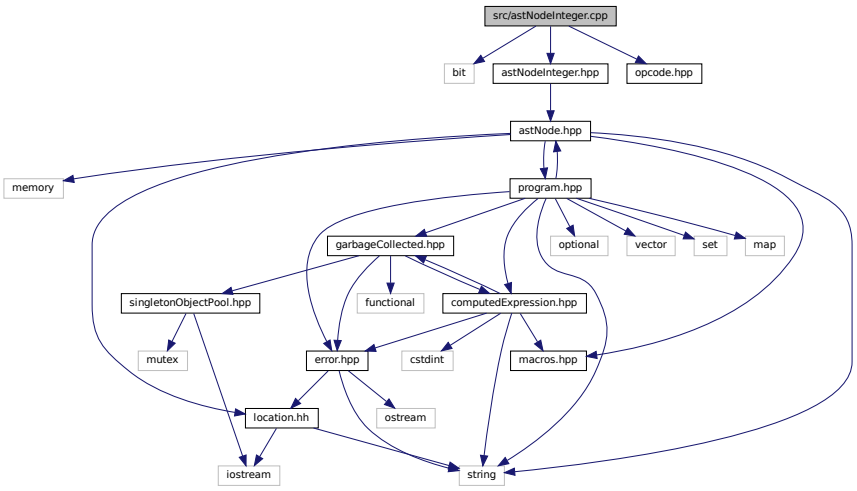
Define the Tang::AstNodeIfElse class.

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:



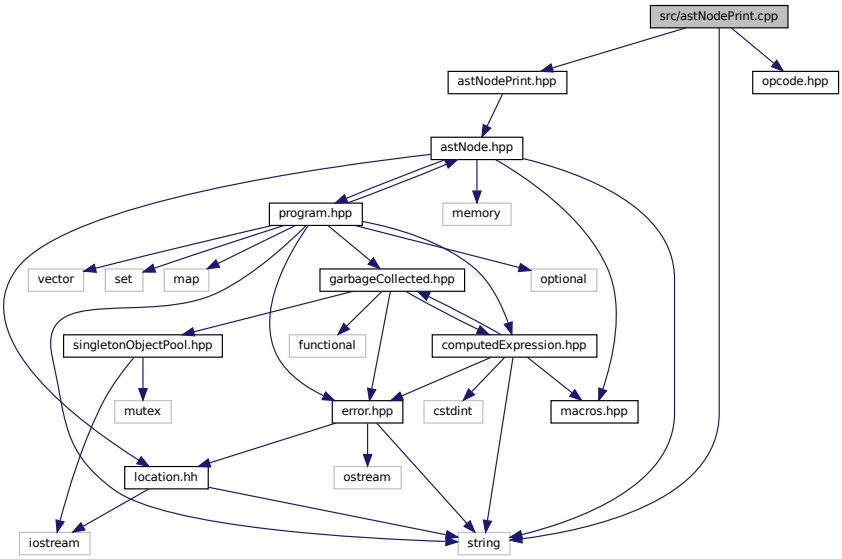
6.51.1 Detailed Description

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

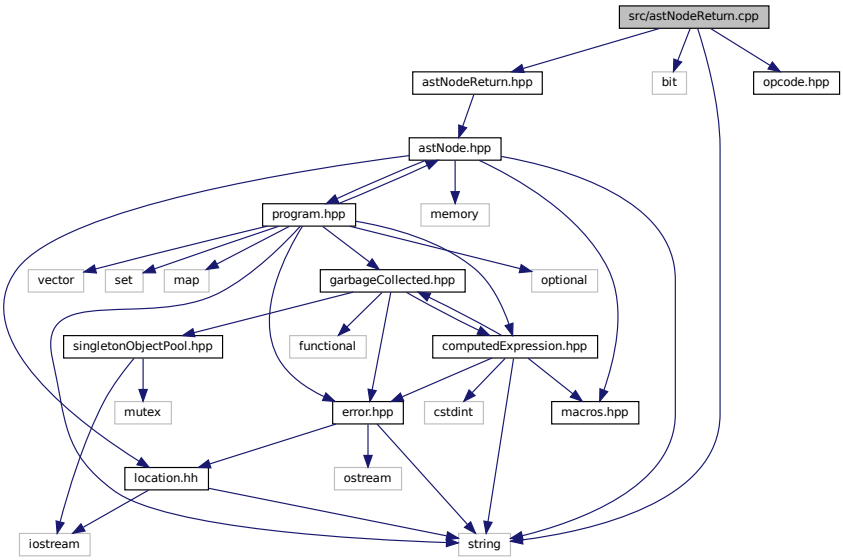
Define the Tang::AstNodePrint class.

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.

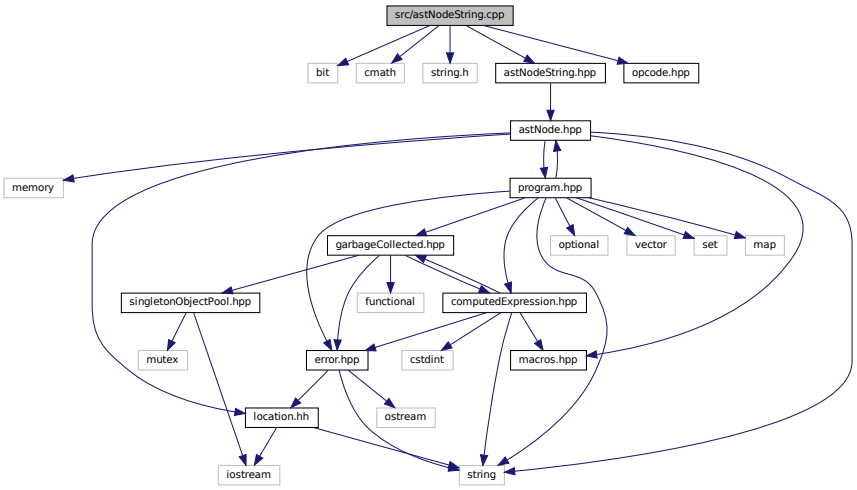
6.54 src/astNodeString.cpp File Reference

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

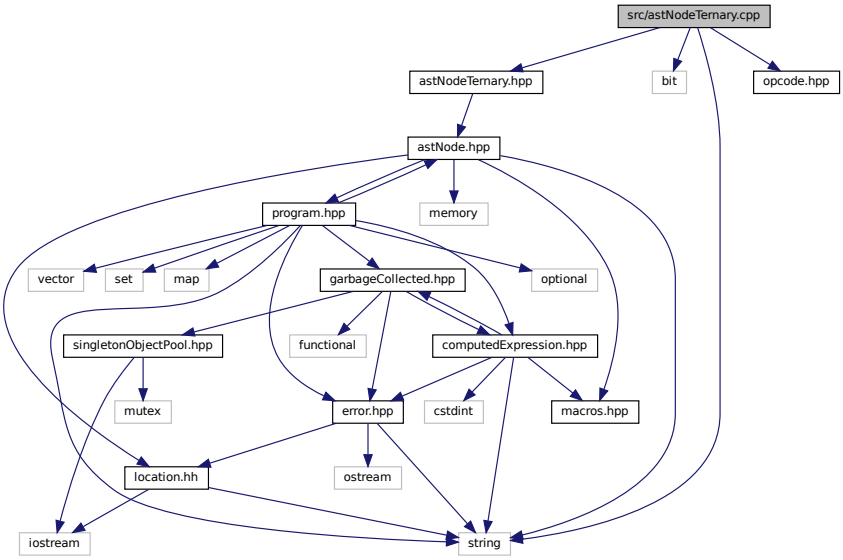
Define the Tang::AstNodeString class.

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

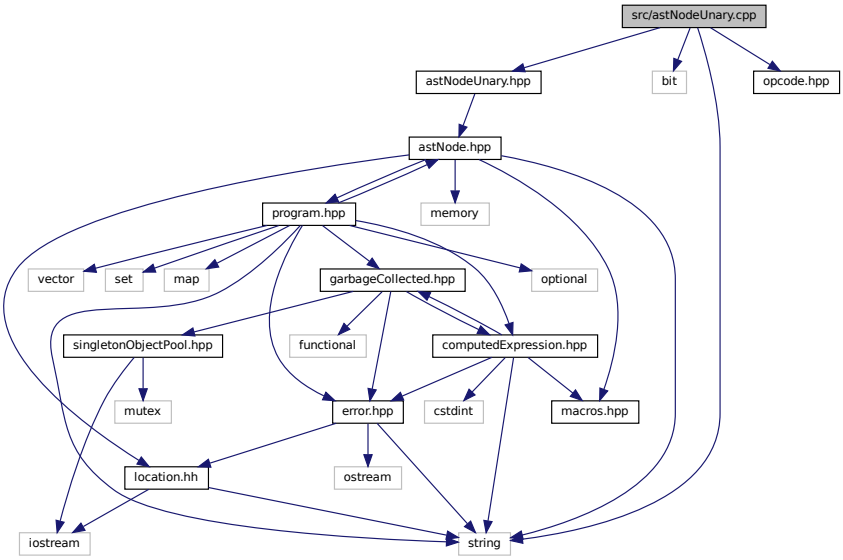
Define the Tang::AstNodeTernary class.

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

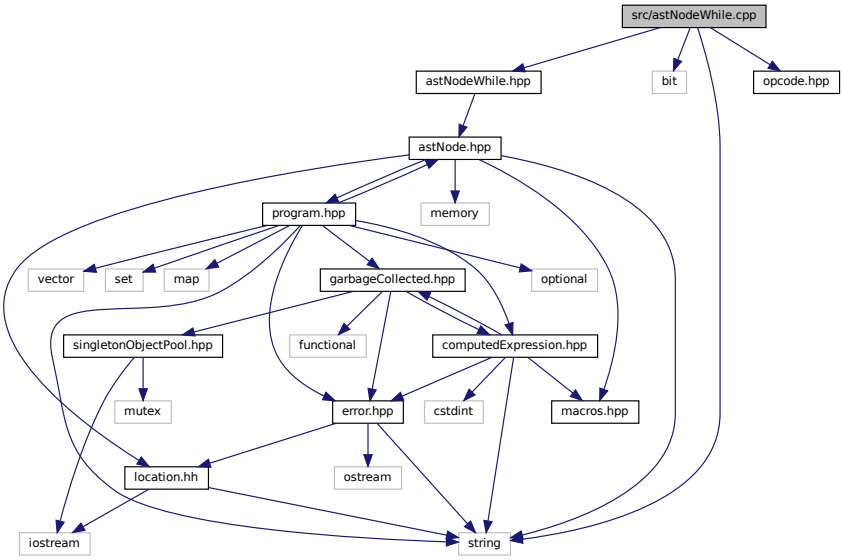
Define the Tang::AstNodeUnary class.

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

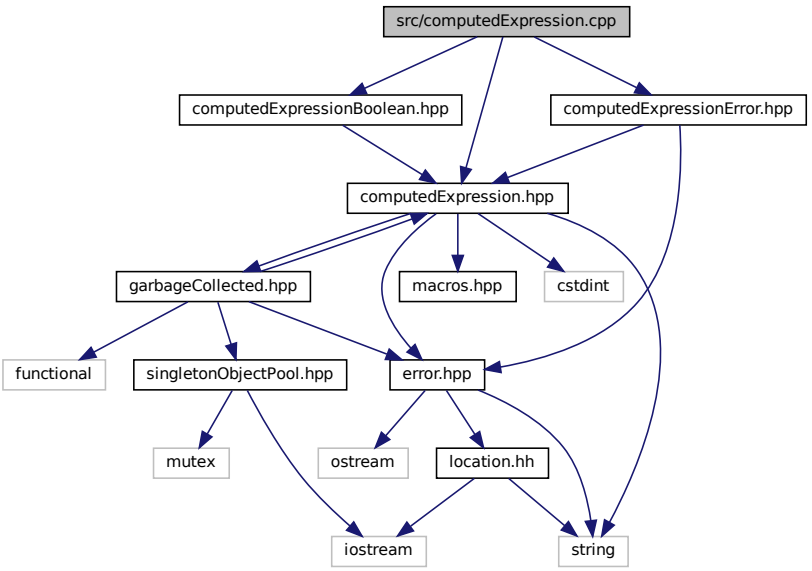
Define the Tang::AstNodeWhile class.

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

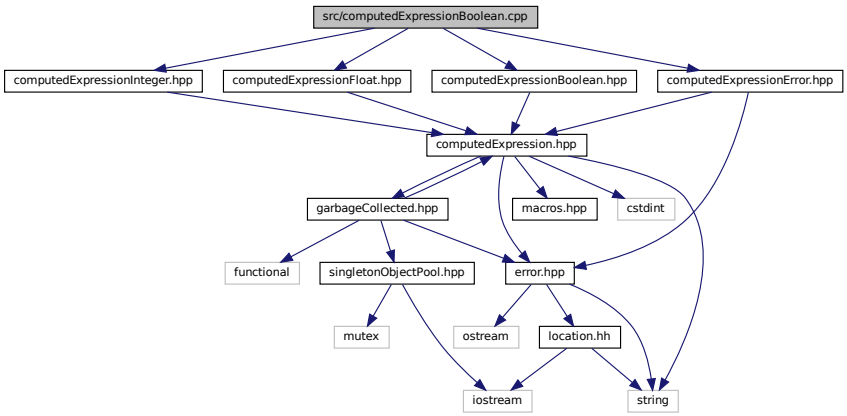
Define the Tang::ComputedExpression class.

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

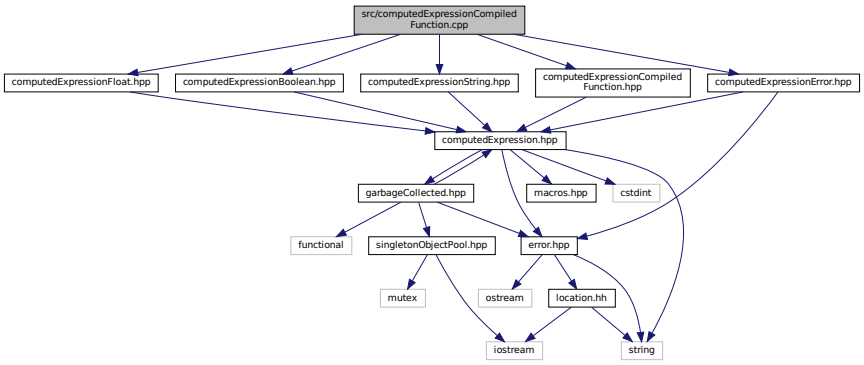
Define the Tang::ComputedExpressionBoolean class.

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

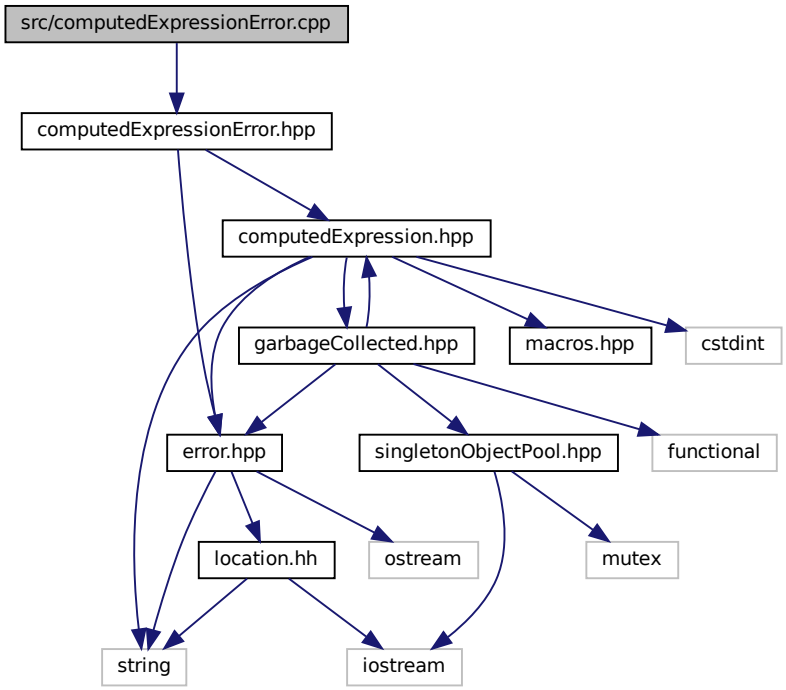
Define the Tang::ComputedExpressionCompiledFunction class.

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

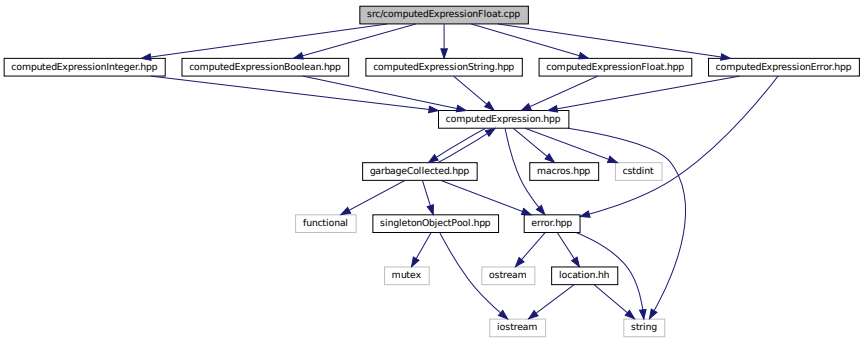
Define the Tang::ComputedExpressionError class.

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

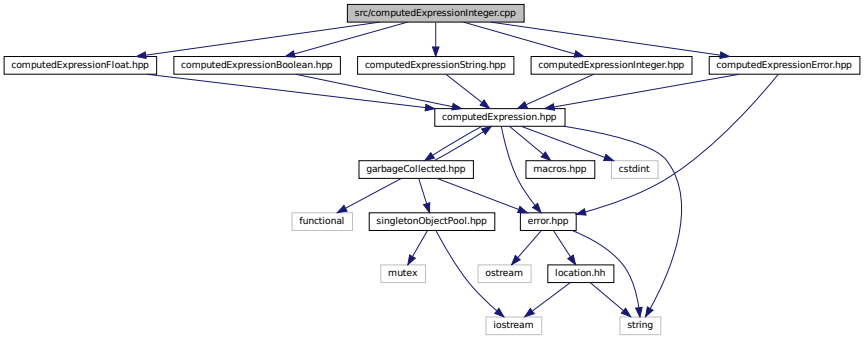
Define the Tang::ComputedExpressionFloat class.

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

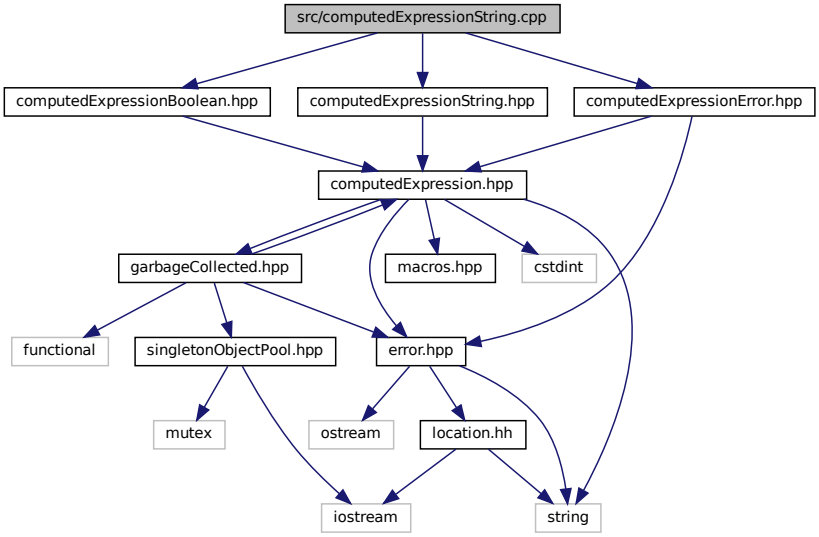
Define the Tang::ComputedExpressionInteger class.

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

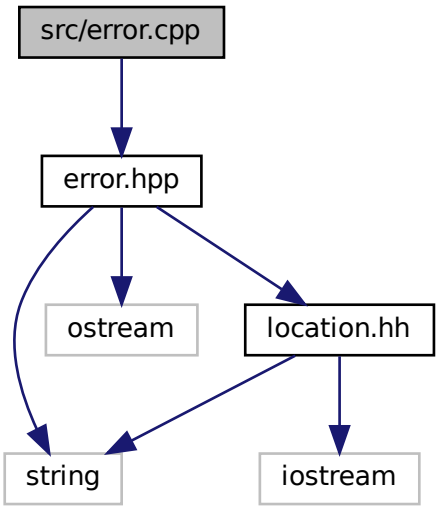
Define the Tang::ComputedExpressionString class.

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

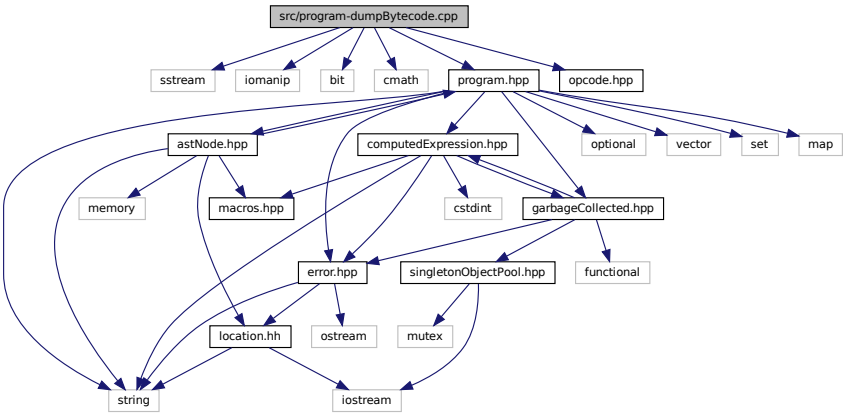
The output stream.

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

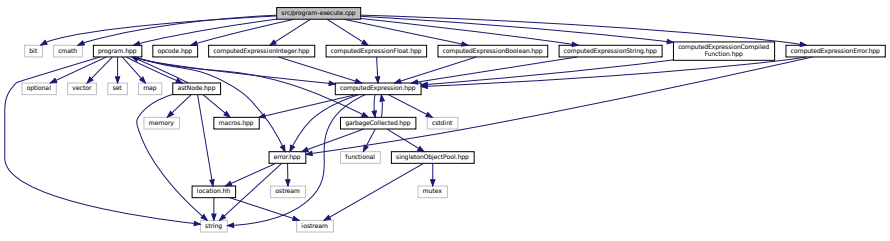
x	The number of additional vector entries that should exist.

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

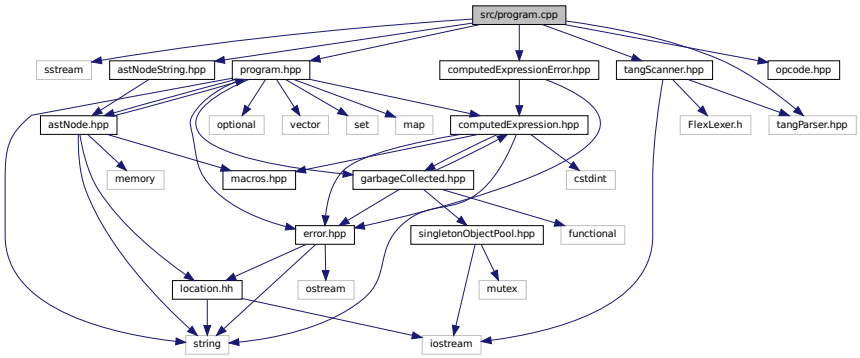
x	The number of entries that should exist in the stack.

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

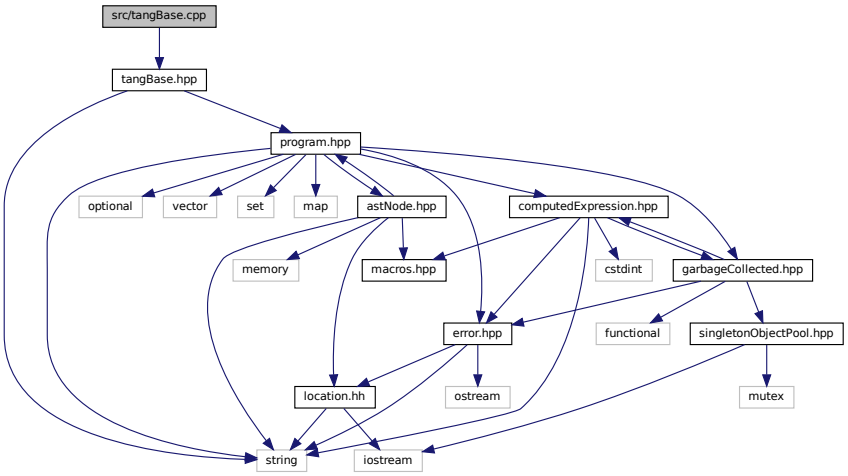
Define the Tang::Program class.

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

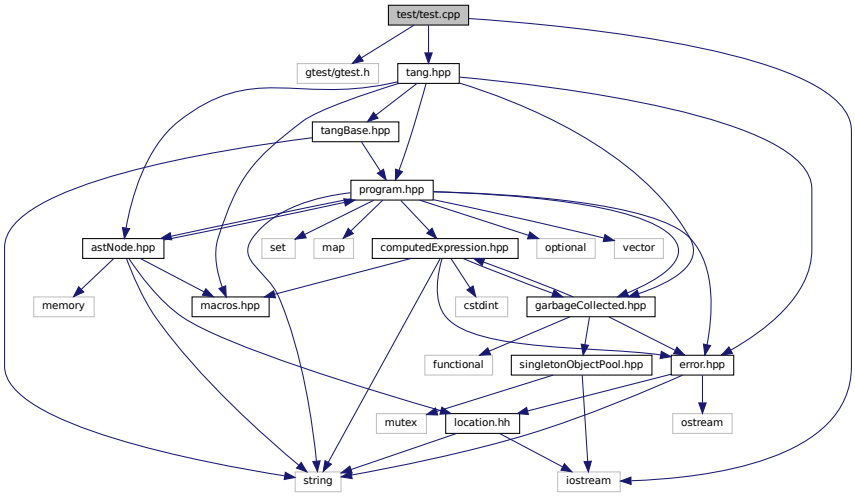
Define the Tang::TangBase class.

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)
- int **main** (int argc, char **argv)

6.70.1 Detailed Description

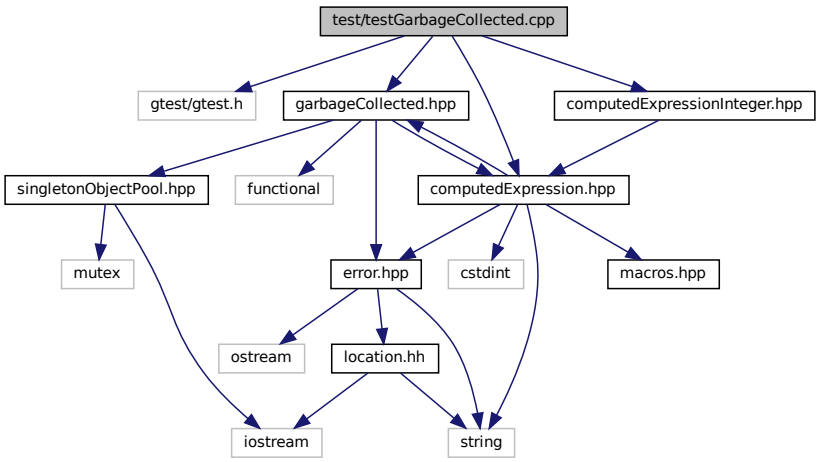
Test the general language behaviors.

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

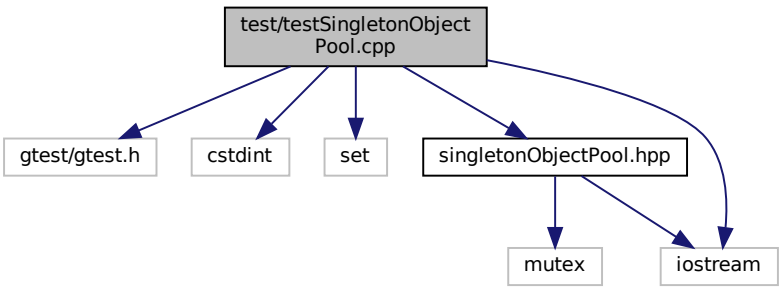
Test the generic behavior of the Tang::GarbageCollected class.

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, 97
Tang::ComputedExpressionBoolean, 107
Tang::ComputedExpressionCompiledFunction, 116
Tang::ComputedExpressionError, 127
Tang::ComputedExpressionFloat, 136
Tang::ComputedExpressionInteger, 146
Tang::ComputedExpressionString, 156
__boolean
Tang::ComputedExpression, 98
Tang::ComputedExpressionBoolean, 107
Tang::ComputedExpressionCompiledFunction, 117
Tang::ComputedExpressionError, 127
Tang::ComputedExpressionFloat, 137
Tang::ComputedExpressionInteger, 146
Tang::ComputedExpressionString, 156
__divide
Tang::ComputedExpression, 98
Tang::ComputedExpressionBoolean, 107
Tang::ComputedExpressionCompiledFunction, 117
Tang::ComputedExpressionError, 127
Tang::ComputedExpressionFloat, 137
Tang::ComputedExpressionInteger, 146
Tang::ComputedExpressionString, 156
__equal
Tang::ComputedExpression, 98
Tang::ComputedExpressionBoolean, 108
Tang::ComputedExpressionCompiledFunction, 117
Tang::ComputedExpressionError, 128
Tang::ComputedExpressionFloat, 137
Tang::ComputedExpressionInteger, 147
Tang::ComputedExpressionString, 157
__float
Tang::ComputedExpression, 99
Tang::ComputedExpressionBoolean, 108
Tang::ComputedExpressionCompiledFunction, 118
Tang::ComputedExpressionError, 128
Tang::ComputedExpressionFloat, 138
Tang::ComputedExpressionInteger, 147
Tang::ComputedExpressionString, 157
__integer
Tang::ComputedExpression, 99
Tang::ComputedExpressionBoolean, 108
Tang::ComputedExpressionCompiledFunction, 118
Tang::ComputedExpressionError, 128
Tang::ComputedExpressionFloat, 138
Tang::ComputedExpressionInteger, 147
Tang::ComputedExpressionString, 157
__lessThan
Tang::ComputedExpression, 99
Tang::ComputedExpressionBoolean, 109
Tang::ComputedExpressionCompiledFunction, 118
Tang::ComputedExpressionError, 129
Tang::ComputedExpressionFloat, 138
Tang::ComputedExpressionInteger, 148
Tang::ComputedExpressionString, 157
__modulo
Tang::ComputedExpression, 100
Tang::ComputedExpressionBoolean, 109
Tang::ComputedExpressionCompiledFunction, 119
Tang::ComputedExpressionError, 129
Tang::ComputedExpressionFloat, 139
Tang::ComputedExpressionInteger, 148
Tang::ComputedExpressionString, 158
__multiply
Tang::ComputedExpression, 100
Tang::ComputedExpressionBoolean, 110
Tang::ComputedExpressionCompiledFunction, 119
Tang::ComputedExpressionError, 129
Tang::ComputedExpressionFloat, 139
Tang::ComputedExpressionInteger, 148
Tang::ComputedExpressionString, 158
__negative
Tang::ComputedExpression, 100
Tang::ComputedExpressionBoolean, 110
Tang::ComputedExpressionCompiledFunction, 119
Tang::ComputedExpressionError, 130
Tang::ComputedExpressionFloat, 139
Tang::ComputedExpressionInteger, 149
Tang::ComputedExpressionString, 159
__not
Tang::ComputedExpression, 101
Tang::ComputedExpressionBoolean, 110
Tang::ComputedExpressionCompiledFunction, 120
Tang::ComputedExpressionError, 130
Tang::ComputedExpressionFloat, 140
Tang::ComputedExpressionInteger, 149
Tang::ComputedExpressionString, 159
__string
Tang::ComputedExpression, 101
Tang::ComputedExpressionBoolean, 110
Tang::ComputedExpressionCompiledFunction, 120
Tang::ComputedExpressionError, 130
Tang::ComputedExpressionFloat, 140
Tang::ComputedExpressionInteger, 149
Tang::ComputedExpressionString, 159
__subtract
Tang::ComputedExpression, 101

Tang::ComputedExpressionBoolean, 111
Tang::ComputedExpressionCompiledFunction, 120
Tang::ComputedExpressionError, 131
Tang::ComputedExpressionFloat, 140
Tang::ComputedExpressionInteger, 150
Tang::ComputedExpressionString, 159
~GarbageCollected
Tang::GarbageCollected, 168

ADD
opcode.hpp, 229
Add
Tang::AstNodeBinary, 22
addBytecode
Tang::Program, 186
addIdentifier
Tang::Program, 188
addString
Tang::Program, 188
And
Tang::AstNodeBinary, 22
AstNode
Tang::AstNode, 13
AstNodeAssign
Tang::AstNodeAssign, 17
AstNodeBinary
Tang::AstNodeBinary, 22
AstNodeBlock
Tang::AstNodeBlock, 26
AstNodeBoolean
Tang::AstNodeBoolean, 30
AstNodeCast
Tang::AstNodeCast, 34
AstNodeDoWhile
Tang::AstNodeDoWhile, 38
AstNodeFloat
Tang::AstNodeFloat, 41
AstNodeFor
Tang::AstNodeFor, 45
AstNodeFunctionCall
Tang::AstNodeFunctionCall, 50
AstNodeFunctionDeclaration
Tang::AstNodeFunctionDeclaration, 54
AstNodeIdentifier
Tang::AstNodeIdentifier, 59
AstNodeIfElse
Tang::AstNodeIfElse, 63
AstNodeInteger
Tang::AstNodeInteger, 68
AstNodePrint
Tang::AstNodePrint, 72
AstNodeReturn
Tang::AstNodeReturn, 76
AstNodeString
Tang::AstNodeString, 79
AstNodeTernary
Tang::AstNodeTernary, 84
AstNodeUnary
Tang::AstNodeUnary, 88
AstNodeWhile
Tang::AstNodeWhile, 93

BOOLEAN
opcode.hpp, 229
Boolean
Tang::AstNodeCast, 34
build/generated/location.hh, 197

CALLFUNC
opcode.hpp, 229
CASTBOOLEAN
opcode.hpp, 229
CASTFLOAT
opcode.hpp, 229
CASTINTEGER
opcode.hpp, 229
CodeType
Tang::Program, 186
collectFunctionDeclarations
Tang::AstNode, 13
Tang::AstNodeAssign, 17
Tang::AstNodeBinary, 23
Tang::AstNodeBlock, 26
Tang::AstNodeBoolean, 30
Tang::AstNodeCast, 34
Tang::AstNodeDoWhile, 38
Tang::AstNodeFloat, 42
Tang::AstNodeFor, 45
Tang::AstNodeFunctionCall, 50
Tang::AstNodeFunctionDeclaration, 55
Tang::AstNodeIdentifier, 59
Tang::AstNodeIfElse, 63
Tang::AstNodeInteger, 68
Tang::AstNodePrint, 72
Tang::AstNodeReturn, 76
Tang::AstNodeString, 80
Tang::AstNodeTernary, 84
Tang::AstNodeUnary, 88
Tang::AstNodeWhile, 93
collectIdentifiers
Tang::AstNode, 14
Tang::AstNodeAssign, 19
Tang::AstNodeBinary, 23
Tang::AstNodeBlock, 27
Tang::AstNodeBoolean, 30
Tang::AstNodeCast, 34
Tang::AstNodeDoWhile, 38
Tang::AstNodeFloat, 42
Tang::AstNodeFor, 46
Tang::AstNodeFunctionCall, 50
Tang::AstNodeFunctionDeclaration, 55
Tang::AstNodeIdentifier, 59
Tang::AstNodeIfElse, 64
Tang::AstNodeInteger, 68
Tang::AstNodePrint, 73
Tang::AstNodeReturn, 76
Tang::AstNodeString, 80
Tang::AstNodeTernary, 84

Tang::AstNodeUnary, 89
Tang::AstNodeWhile, 93
collectStrings
Tang::AstNode, 14
Tang::AstNodeAssign, 19
Tang::AstNodeBinary, 23
Tang::AstNodeBlock, 27
Tang::AstNodeBoolean, 31
Tang::AstNodeCast, 35
Tang::AstNodeDoWhile, 39
Tang::AstNodeFloat, 42
Tang::AstNodeFor, 46
Tang::AstNodeFunctionCall, 52
Tang::AstNodeFunctionDeclaration, 56
Tang::AstNodeIdentifier, 60
Tang::AstNodeIfElse, 64
Tang::AstNodeInteger, 69
Tang::AstNodePrint, 73
Tang::AstNodeReturn, 77
Tang::AstNodeString, 80
Tang::AstNodeTernary, 85
Tang::AstNodeUnary, 89
Tang::AstNodeWhile, 94
compile
Tang::AstNode, 14
Tang::AstNodeAssign, 19
Tang::AstNodeBinary, 24
Tang::AstNodeBlock, 27
Tang::AstNodeBoolean, 31
Tang::AstNodeCast, 35
Tang::AstNodeDoWhile, 39
Tang::AstNodeFloat, 43
Tang::AstNodeFor, 46
Tang::AstNodeFunctionCall, 52
Tang::AstNodeFunctionDeclaration, 56
Tang::AstNodeIdentifier, 60
Tang::AstNodeIfElse, 64
Tang::AstNodeInteger, 69
Tang::AstNodePrint, 73
Tang::AstNodeReturn, 77
Tang::AstNodeString, 81
Tang::AstNodeTernary, 85
Tang::AstNodeUnary, 89
Tang::AstNodeWhile, 94
compileLiteral
Tang::AstNodeString, 81
compileScript
Tang::TangBase, 194
ComputedExpressionBoolean
Tang::ComputedExpressionBoolean, 107
ComputedExpressionCompiledFunction
Tang::ComputedExpressionCompiledFunction, 116
ComputedExpressionError
Tang::ComputedExpressionError, 126
ComputedExpressionFloat
Tang::ComputedExpressionFloat, 136
ComputedExpressionInteger
Tang::ComputedExpressionInteger, 145
ComputedExpressionString
Tang::ComputedExpressionString, 155

Default
Tang::AstNodePrint, 72
DIVIDE
opcode.hpp, 229
Divide
Tang::AstNodeBinary, 22
dump
Tang::AstNode, 15
Tang::AstNodeAssign, 20
Tang::AstNodeBinary, 24
Tang::AstNodeBlock, 28
Tang::AstNodeBoolean, 31
Tang::AstNodeCast, 36
Tang::AstNodeDoWhile, 39
Tang::AstNodeFloat, 43
Tang::AstNodeFor, 48
Tang::AstNodeFunctionCall, 52
Tang::AstNodeFunctionDeclaration, 56
Tang::AstNodeIdentifier, 61
Tang::AstNodeIfElse, 66
Tang::AstNodeInteger, 70
Tang::AstNodePrint, 74
Tang::AstNodeReturn, 77
Tang::AstNodeString, 82
Tang::AstNodeTernary, 86
Tang::AstNodeUnary, 91
Tang::AstNodeWhile, 95
Tang::ComputedExpression, 102
Tang::ComputedExpressionBoolean, 111
Tang::ComputedExpressionCompiledFunction, 121
Tang::ComputedExpressionError, 131
Tang::ComputedExpressionFloat, 141
Tang::ComputedExpressionInteger, 150
Tang::ComputedExpressionString, 160
dumpBytecode
Tang::Program, 188
DUMPPROGRAMCHECK
program-dumpBytecode.cpp, 257

EQ
opcode.hpp, 229
Equal
Tang::AstNodeBinary, 22
Error
Tang::Error, 164
error.cpp
operator<<, 256
execute
Tang::Program, 188
EXECUTEPROGRAMCHECK
program-execute.cpp, 259

FLOAT
opcode.hpp, 229
Float
Tang::AstNodeCast, 34

FUNCTION
opcode.hpp, 229
functionsDeclared
Tang::Program, 191

GarbageCollected
Tang::GarbageCollected, 167, 168
get
Tang::SingletonObjectPool< T >, 192
get_next_token
Tang::TangScanner, 196
getAst
Tang::Program, 189
getBytecode
Tang::Program, 189
getCode
Tang::Program, 189
getIdentifiers
Tang::Program, 190
getInstance
Tang::SingletonObjectPool< T >, 193
getResult
Tang::Program, 190
getStrings
Tang::Program, 190
GreaterThan
Tang::AstNodeBinary, 22
GreaterThanEqual
Tang::AstNodeBinary, 22
GT
opcode.hpp, 229
GTE
opcode.hpp, 229

include/astNode.hpp, 199
include/astNodeAssign.hpp, 200
include/astNodeBinary.hpp, 201
include/astNodeBlock.hpp, 202
include/astNodeBoolean.hpp, 203
include/astNodeCast.hpp, 204
include/astNodeDoWhile.hpp, 205
include/astNodeFloat.hpp, 206
include/astNodeFor.hpp, 207
include/astNodeFunctionCall.hpp, 208
include/astNodeFunctionDeclaration.hpp, 209
include/astNodeIdentifier.hpp, 210
include/astNodeIfElse.hpp, 211
include/astNodeInteger.hpp, 212
include/astNodePrint.hpp, 213
include/astNodeReturn.hpp, 214
include/astNodeString.hpp, 215
include/astNodeTernary.hpp, 216
include/astNodeUnary.hpp, 217
include/astNodeWhile.hpp, 218
include/computedExpression.hpp, 219
include/computedExpressionBoolean.hpp, 220
include/computedExpressionCompiledFunction.hpp, 221
include/computedExpressionError.hpp, 222
include/computedExpressionFloat.hpp, 223
include/computedExpressionInteger.hpp, 224
include/computedExpressionString.hpp, 225
include/error.hpp, 226
include/garbageCollected.hpp, 227
include/macros.hpp, 227
include/opcode.hpp, 228
include/program.hpp, 229
include/singletonObjectPool.hpp, 230
include/tang.hpp, 231
include/tangBase.hpp, 232
include/tangScanner.hpp, 234
INTEGER
opcode.hpp, 229
Integer
Tang::AstNodeCast, 34
is_equal
Tang::ComputedExpression, 102–104
Tang::ComputedExpressionBoolean, 111–113
Tang::ComputedExpressionCompiledFunction, 121, 123, 124
Tang::ComputedExpressionError, 131–133
Tang::ComputedExpressionFloat, 141–143
Tang::ComputedExpressionInteger, 151, 152
Tang::ComputedExpressionString, 160–162

JMP
opcode.hpp, 229
JMPF
opcode.hpp, 229
JMPF_POP
opcode.hpp, 229
JMPT
opcode.hpp, 229
JMPT_POP
opcode.hpp, 229

LessThan
Tang::AstNodeBinary, 22
LessThanEqual
Tang::AstNodeBinary, 22
location.hh
operator<<, 198, 199
LT
opcode.hpp, 229
LTE
opcode.hpp, 229

make
Tang::GarbageCollected, 168
makeCopy
Tang::ComputedExpression, 104
Tang::ComputedExpressionBoolean, 114
Tang::ComputedExpressionCompiledFunction, 124
Tang::ComputedExpressionError, 134
Tang::ComputedExpressionFloat, 143
Tang::ComputedExpressionInteger, 153
Tang::ComputedExpressionString, 162
MODULO

opcode.hpp, 229
Modulo
Tang::AstNodeBinary, 22
MULTIPLY
opcode.hpp, 229
Multiply
Tang::AstNodeBinary, 22

NEGATIVE
opcode.hpp, 229
Negative
Tang::AstNodeUnary, 88
NEQ
opcode.hpp, 229
NOT
opcode.hpp, 229
Not
Tang::AstNodeUnary, 88
NotEqual
Tang::AstNodeBinary, 22
NULLVAL
opcode.hpp, 229

Opcode
opcode.hpp, 228
opcode.hpp
ADD, 229
BOOLEAN, 229
CALLFUNC, 229
CASTBOOLEAN, 229
CASTFLOAT, 229
CASTINTEGER, 229
DIVIDE, 229
EQ, 229
FLOAT, 229
FUNCTION, 229
GT, 229
GTE, 229
INTEGER, 229
JMP, 229
JMPF, 229
JMPF_POP, 229
JMPT, 229
JMPT_POP, 229
LT, 229
LTE, 229
MODULO, 229
MULTIPLY, 229
NEGATIVE, 229
NEQ, 229
NOT, 229
NULLVAL, 229
Opcode, 228
PEEK, 229
POKE, 229
POP, 229
PRINT, 229
RETURN, 229
STRING, 229
SUBTRACT, 229
Operation
Tang::AstNodeBinary, 22
Operator
Tang::AstNodeUnary, 88
operator!
Tang::GarbageCollected, 169
operator!=
Tang::GarbageCollected, 169
operator<
Tang::GarbageCollected, 174
operator<<
error.cpp, 256
location.hh, 198, 199
Tang::Error, 164
Tang::GarbageCollected, 181
operator<=
Tang::GarbageCollected, 174
operator>
Tang::GarbageCollected, 180
operator>=
Tang::GarbageCollected, 180
operator*
Tang::GarbageCollected, 170, 171
operator+
Tang::GarbageCollected, 171
operator-
Tang::GarbageCollected, 172
operator->
Tang::GarbageCollected, 173
operator/
Tang::GarbageCollected, 173
operator=
Tang::GarbageCollected, 175
operator==
Tang::GarbageCollected, 176–178
operator%
Tang::GarbageCollected, 170
Or
Tang::AstNodeBinary, 22

PEEK
opcode.hpp, 229
POKE
opcode.hpp, 229
POP
opcode.hpp, 229
PRINT
opcode.hpp, 229
Program
Tang::Program, 186
program-dumpBytecode.cpp
DUMPPROGRAMCHECK, 257
program-execute.cpp
EXECUTEPROGRAMCHECK, 259
STACKCHECK, 259
pushEnvironment
Tang::Program, 190

recycle
Tang::SingletonObjectPool< T >, 193
RETURN
opcode.hpp, 229

Script
Tang::Program, 186
setFunctionStackDeclaration
Tang::Program, 191
setJumpTarget
Tang::Program, 191
src/astNode.cpp, 235
src/astNodeAssign.cpp, 235
src/astNodeBinary.cpp, 236
src/astNodeBlock.cpp, 237
src/astNodeBoolean.cpp, 237
src/astNodeCast.cpp, 238
src/astNodeDoWhile.cpp, 239
src/astNodeFloat.cpp, 239
src/astNodeFor.cpp, 240
src/astNodeFunctionCall.cpp, 241
src/astNodeFunctionDeclaration.cpp, 242
src/astNodeIdentifier.cpp, 243
src/astNodeIfElse.cpp, 244
src/astNodeInteger.cpp, 244
src/astNodePrint.cpp, 245
src/astNodeReturn.cpp, 246
src/astNodeString.cpp, 246
src/astNodeTernary.cpp, 247
src/astNodeUnary.cpp, 248
src/astNodeWhile.cpp, 249
src/computedExpression.cpp, 250
src/computedExpressionBoolean.cpp, 251
src/computedExpressionCompiledFunction.cpp, 252
src/computedExpressionError.cpp, 252
src/computedExpressionFloat.cpp, 253
src/computedExpressionInteger.cpp, 254
src/computedExpressionString.cpp, 255
src/error.cpp, 255
src/program-dumpBytecode.cpp, 257
src/program-execute.cpp, 258
src/program.cpp, 259
src/tangBase.cpp, 260
STACKCHECK
program-execute.cpp, 259
STRING
opcode.hpp, 229
SUBTRACT
opcode.hpp, 229
Subtract
Tang::AstNodeBinary, 22

Tang::AstNode, 11
AstNode, 13
collectFunctionDeclarations, 13
collectIdentifiers, 14
collectStrings, 14
compile, 14
dump, 15
Tang::AstNodeAssign, 16
AstNodeAssign, 17
collectFunctionDeclarations, 17
collectIdentifiers, 19
collectStrings, 19
compile, 19
dump, 20
Tang::AstNodeBinary, 20
Add, 22
And, 22
AstNodeBinary, 22
collectFunctionDeclarations, 23
collectIdentifiers, 23
collectStrings, 23
compile, 24
Divide, 22
dump, 24
Equal, 22
GreaterThan, 22
GreaterThanEqual, 22
LessThan, 22
LessThanEqual, 22
Modulo, 22
Multiply, 22
NotEqual, 22
Operation, 22
Or, 22
Subtract, 22
Tang::AstNodeBlock, 25
AstNodeBlock, 26
collectFunctionDeclarations, 26
collectIdentifiers, 27
collectStrings, 27
compile, 27
dump, 28
Tang::AstNodeBoolean, 28
AstNodeBoolean, 30
collectFunctionDeclarations, 30
collectIdentifiers, 30
collectStrings, 31
compile, 31
dump, 31
Tang::AstNodeCast, 32
AstNodeCast, 34
Boolean, 34
collectFunctionDeclarations, 34
collectIdentifiers, 34
collectStrings, 35
compile, 35
dump, 36
Float, 34
Integer, 34
Type, 33
Tang::AstNodeDoWhile, 36
AstNodeDoWhile, 38
collectFunctionDeclarations, 38
collectIdentifiers, 38
collectStrings, 39

compile, 39
dump, 39
Tang::AstNodeFloat, 40
AstNodeFloat, 41
collectFunctionDeclarations, 42
collectIdentifiers, 42
collectStrings, 42
compile, 43
dump, 43
Tang::AstNodeFor, 44
AstNodeFor, 45
collectFunctionDeclarations, 45
collectIdentifiers, 46
collectStrings, 46
compile, 46
dump, 48
Tang::AstNodeFunctionCall, 49
AstNodeFunctionCall, 50
collectFunctionDeclarations, 50
collectIdentifiers, 50
collectStrings, 52
compile, 52
dump, 52
Tang::AstNodeFunctionDeclaration, 53
AstNodeFunctionDeclaration, 54
collectFunctionDeclarations, 55
collectIdentifiers, 55
collectStrings, 56
compile, 56
dump, 56
Tang::AstNodeIdentifier, 57
AstNodeIdentifier, 59
collectFunctionDeclarations, 59
collectIdentifiers, 59
collectStrings, 60
compile, 60
dump, 61
Tang::AstNodeIfElse, 61
AstNodeIfElse, 63
collectFunctionDeclarations, 63
collectIdentifiers, 64
collectStrings, 64
compile, 64
dump, 66
Tang::AstNodeInteger, 67
AstNodeInteger, 68
collectFunctionDeclarations, 68
collectIdentifiers, 68
collectStrings, 69
compile, 69
dump, 70
Tang::AstNodePrint, 70
AstNodePrint, 72
collectFunctionDeclarations, 72
collectIdentifiers, 73
collectStrings, 73
compile, 73
Default, 72
dump, 74
Type, 72
Tang::AstNodeReturn, 74
AstNodeReturn, 76
collectFunctionDeclarations, 76
collectIdentifiers, 76
collectStrings, 77
compile, 77
dump, 77
Tang::AstNodeString, 78
AstNodeString, 79
collectFunctionDeclarations, 80
collectIdentifiers, 80
collectStrings, 80
compile, 81
compileLiteral, 81
dump, 82
Tang::AstNodeTernary, 83
AstNodeTernary, 84
collectFunctionDeclarations, 84
collectIdentifiers, 84
collectStrings, 85
compile, 85
dump, 86
Tang::AstNodeUnary, 86
AstNodeUnary, 88
collectFunctionDeclarations, 88
collectIdentifiers, 89
collectStrings, 89
compile, 89
dump, 91
Negative, 88
Not, 88
Operator, 88
Tang::AstNodeWhile, 91
AstNodeWhile, 93
collectFunctionDeclarations, 93
collectIdentifiers, 93
collectStrings, 94
compile, 94
dump, 95
Tang::ComputedExpression, 95
__add, 97
__boolean, 98
__divide, 98
__equal, 98
__float, 99
__integer, 99
__lessThan, 99
__modulo, 100
__multiply, 100
__negative, 100
__not, 101
__string, 101
__subtract, 101
dump, 102
is_equal, 102–104
makeCopy, 104

Tang::ComputedExpressionBoolean, 105
__add, 107
__boolean, 107
__divide, 107
__equal, 108
__float, 108
__integer, 108
__lessThan, 109
__modulo, 109
__multiply, 110
__negative, 110
__not, 110
__string, 110
__subtract, 111
ComputedExpressionBoolean, 107
dump, 111
is_equal, 111–113
makeCopy, 114
Tang::ComputedExpressionCompiledFunction, 114
__add, 116
__boolean, 117
__divide, 117
__equal, 117
__float, 118
__integer, 118
__lessThan, 118
__modulo, 119
__multiply, 119
__negative, 119
__not, 120
__string, 120
__subtract, 120
ComputedExpressionCompiledFunction, 116
dump, 121
is_equal, 121, 123, 124
makeCopy, 124
Tang::ComputedExpressionError, 125
__add, 127
__boolean, 127
__divide, 127
__equal, 128
__float, 128
__integer, 128
__lessThan, 129
__modulo, 129
__multiply, 129
__negative, 130
__not, 130
__string, 130
__subtract, 131
ComputedExpressionError, 126
dump, 131
is_equal, 131–133
makeCopy, 134
Tang::ComputedExpressionFloat, 134
__add, 136
__boolean, 137
__divide, 137
__equal, 137
__float, 138
__integer, 138
__lessThan, 138
__modulo, 139
__multiply, 139
__negative, 139
__not, 140
__string, 140
__subtract, 140
ComputedExpressionFloat, 136
dump, 141
is_equal, 141–143
makeCopy, 143
Tang::ComputedExpressionInteger, 144
__add, 146
__boolean, 146
__divide, 146
__equal, 147
__float, 147
__integer, 147
__lessThan, 148
__modulo, 148
__multiply, 148
__negative, 149
__not, 149
__string, 149
__subtract, 150
ComputedExpressionInteger, 145
dump, 150
is_equal, 151, 152
makeCopy, 153
Tang::ComputedExpressionString, 153
__add, 156
__boolean, 156
__divide, 156
__equal, 157
__float, 157
__integer, 157
__lessThan, 157
__modulo, 158
__multiply, 158
__negative, 159
__not, 159
__string, 159
__subtract, 159
ComputedExpressionString, 155
dump, 160
is_equal, 160–162
makeCopy, 162
Tang::Error, 163
Error, 164
operator<<, 164
Tang::GarbageCollected, 165
~GarbageCollected, 168
GarbageCollected, 167, 168
make, 168
operator!, 169

operator!=, 169
operator<, 174
operator<<, 181
operator<=, 174
operator>, 180
operator>=, 180
operator*, 170, 171
operator+, 171
operator-, 172
operator->, 173
operator/, 173
operator=, 175
operator==, 176–178
operator%, 170
Tang::location, 181
Tang::position, 183
Tang::Program, 184
addBytecode, 186
addIdentifier, 188
addString, 188
CodeType, 186
dumpBytecode, 188
execute, 188
functionsDeclared, 191
getAst, 189
getBytecode, 189
getCode, 189
getIdentifiers, 190
getResult, 190
getStrings, 190
Program, 186
pushEnvironment, 190
Script, 186
setFunctionStackDeclaration, 191
setJumpTarget, 191
Template, 186
Tang::SingletonObjectPool< T >, 192
get, 192
getInstance, 193
recycle, 193
Tang::TangBase, 193
compileScript, 194
TangBase, 194
Tang::TangScanner, 195
get_next_token, 196
TangScanner, 196
TangBase
Tang::TangBase, 194
TangScanner
Tang::TangScanner, 196
Template
Tang::Program, 186
test/test.cpp, 261
test/testGarbageCollected.cpp, 262
test/testSingletonObjectPool.cpp, 263
Type
Tang::AstNodeCast, 33
Tang::AstNodePrint, 72