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 Member Enumeration Documentation . . . 13
5.1.2.1 PreprocessState . . . 13
5.1.3 Constructor & Destructor Documentation . . . 13
5.1.3.1 AstNode() . . . 13
5.1.4 Member Function Documentation . . . 14
5.1.4.1 compile() . . . 14
5.1.4.2 compilePreprocess() . . . 14
5.1.4.3 dump() . . . 15
5.2 Tang::AstNodeArray Class Reference . . . 15
5.2.1 Detailed Description . . . 16
5.2.2 Member Enumeration Documentation . . . 17
5.2.2.1 PreprocessState . . . 17
5.2.3 Constructor & Destructor Documentation . . . 17
5.2.3.1 AstNodeArray() . . . 17
5.2.4 Member Function Documentation . . . 17
5.2.4.1 compile() . . . 17
5.2.4.2 compilePreprocess() . . . 18
5.2.4.3 dump() . . . 18
5.3 Tang::AstNodeAssign Class Reference . . . 19
5.3.1 Detailed Description . . . 20
5.3.2 Member Enumeration Documentation . . . 20
5.3.2.1 PreprocessState . . . 20
5.3.3 Constructor & Destructor Documentation . . . 20
5.3.3.1 AstNodeAssign() . . . 20
5.3.4 Member Function Documentation . . . 21
5.3.4.1 compile() . . . 21
5.3.4.2 compilePreprocess() . . . 21

5.3.4.3 dump() . . . 22
5.4 Tang::AstNodeBinary Class Reference . . . 22
5.4.1 Detailed Description . . . 23
5.4.2 Member Enumeration Documentation . . . 23
5.4.2.1 Operation . . . 23
5.4.2.2 PreprocessState . . . 24
5.4.3 Constructor & Destructor Documentation . . . 24
5.4.3.1 AstNodeBinary() . . . 24
5.4.4 Member Function Documentation . . . 25
5.4.4.1 compile() . . . 25
5.4.4.2 compilePreprocess() . . . 25
5.4.4.3 dump() . . . 26
5.5 Tang::AstNodeBlock Class Reference . . . 26
5.5.1 Detailed Description . . . 27
5.5.2 Member Enumeration Documentation . . . 27
5.5.2.1 PreprocessState . . . 27
5.5.3 Constructor & Destructor Documentation . . . 28
5.5.3.1 AstNodeBlock() . . . 28
5.5.4 Member Function Documentation . . . 28
5.5.4.1 compile() . . . 28
5.5.4.2 compilePreprocess() . . . 29
5.5.4.3 dump() . . . 29
5.6 Tang::AstNodeBoolean Class Reference . . . 30
5.6.1 Detailed Description . . . 31
5.6.2 Member Enumeration Documentation . . . 31
5.6.2.1 PreprocessState . . . 31
5.6.3 Constructor & Destructor Documentation . . . 31
5.6.3.1 AstNodeBoolean() . . . 31
5.6.4 Member Function Documentation . . . 31
5.6.4.1 compile() . . . 31
5.6.4.2 compilePreprocess() . . . 33
5.6.4.3 dump() . . . 33
5.7 Tang::AstNodeBreak Class Reference . . . 34
5.7.1 Detailed Description . . . 35
5.7.2 Member Enumeration Documentation . . . 35
5.7.2.1 PreprocessState . . . 35
5.7.3 Constructor & Destructor Documentation . . . 35
5.7.3.1 AstNodeBreak() . . . 35
5.7.4 Member Function Documentation . . . 36
5.7.4.1 compile() . . . 36
5.7.4.2 compilePreprocess() . . . 36
5.7.4.3 dump() . . . 37

5.8 Tang::AstNodeCast Class Reference . . . 37
5.8.1 Detailed Description . . . 38
5.8.2 Member Enumeration Documentation . . . 38
5.8.2.1 PreprocessState . . . 38
5.8.2.2 Type . . . 39
5.8.3 Constructor & Destructor Documentation . . . 39
5.8.3.1 AstNodeCast() . . . 39
5.8.4 Member Function Documentation . . . 39
5.8.4.1 compile() . . . 39
5.8.4.2 compilePreprocess() . . . 40
5.8.4.3 dump() . . . 40
5.9 Tang::AstNodeContinue Class Reference . . . 41
5.9.1 Detailed Description . . . 42
5.9.2 Member Enumeration Documentation . . . 42
5.9.2.1 PreprocessState . . . 42
5.9.3 Constructor & Destructor Documentation . . . 42
5.9.3.1 AstNodeContinue() . . . 42
5.9.4 Member Function Documentation . . . 43
5.9.4.1 compile() . . . 43
5.9.4.2 compilePreprocess() . . . 43
5.9.4.3 dump() . . . 44
5.10 Tang::AstNodeDoWhile Class Reference . . . 44
5.10.1 Detailed Description . . . 45
5.10.2 Member Enumeration Documentation . . . 45
5.10.2.1 PreprocessState . . . 45
5.10.3 Constructor & Destructor Documentation . . . 46
5.10.3.1 AstNodeDoWhile() . . . 46
5.10.4 Member Function Documentation . . . 46
5.10.4.1 compile() . . . 46
5.10.4.2 compilePreprocess() . . . 47
5.10.4.3 dump() . . . 47
5.11 Tang::AstNodeFloat Class Reference . . . 48
5.11.1 Detailed Description . . . 49
5.11.2 Member Enumeration Documentation . . . 49
5.11.2.1 PreprocessState . . . 49
5.11.3 Constructor & Destructor Documentation . . . 49
5.11.3.1 AstNodeFloat() . . . 49
5.11.4 Member Function Documentation . . . 50
5.11.4.1 compile() . . . 50
5.11.4.2 compilePreprocess() . . . 50
5.11.4.3 dump() . . . 51
5.12 Tang::AstNodeFor Class Reference . . . 51

5.12.1 Detailed Description . . . 52
5.12.2 Member Enumeration Documentation . . . 52
5.12.2.1 PreprocessState . . . 52
5.12.3 Constructor & Destructor Documentation . . . 53
5.12.3.1 AstNodeFor() . . . 53
5.12.4 Member Function Documentation . . . 53
5.12.4.1 compile() . . . 53
5.12.4.2 compilePreprocess() . . . 54
5.12.4.3 dump() . . . 54
5.13 Tang::AstNodeFunctionCall Class Reference . . . 55
5.13.1 Detailed Description . . . 56
5.13.2 Member Enumeration Documentation . . . 56
5.13.2.1 PreprocessState . . . 56
5.13.3 Constructor & Destructor Documentation . . . 56
5.13.3.1 AstNodeFunctionCall() . . . 56
5.13.4 Member Function Documentation . . . 57
5.13.4.1 compile() . . . 57
5.13.4.2 compilePreprocess() . . . 57
5.13.4.3 dump() . . . 58
5.14 Tang::AstNodeFunctionDeclaration Class Reference . . . 58
5.14.1 Detailed Description . . . 59
5.14.2 Member Enumeration Documentation . . . 59
5.14.2.1 PreprocessState . . . 59
5.14.3 Constructor & Destructor Documentation . . . 60
5.14.3.1 AstNodeFunctionDeclaration() . . . 60
5.14.4 Member Function Documentation . . . 60
5.14.4.1 compile() . . . 60
5.14.4.2 compilePreprocess() . . . 61
5.14.4.3 dump() . . . 62
5.15 Tang::AstNodeIdentifier Class Reference . . . 62
5.15.1 Detailed Description . . . 63
5.15.2 Member Enumeration Documentation . . . 64
5.15.2.1 PreprocessState . . . 64
5.15.3 Constructor & Destructor Documentation . . . 64
5.15.3.1 AstNodeIdentifier() . . . 64
5.15.4 Member Function Documentation . . . 64
5.15.4.1 compile() . . . 64
5.15.4.2 compilePreprocess() . . . 65
5.15.4.3 dump() . . . 66
5.16 Tang::AstNodeIfElse Class Reference . . . 66
5.16.1 Detailed Description . . . 67
5.16.2 Member Enumeration Documentation . . . 67

5.16.2.1 PreprocessState . . . 67
5.16.3 Constructor & Destructor Documentation . . . 68
5.16.3.1 AstNodeIfElse() [1/2] . . . 68
5.16.3.2 AstNodeIfElse() [2/2] . . . 68
5.16.4 Member Function Documentation . . . 68
5.16.4.1 compile() . . . 69
5.16.4.2 compilePreprocess() . . . 69
5.16.4.3 dump() . . . 69
5.17 Tang::AstNodeIndex Class Reference . . . 70
5.17.1 Detailed Description . . . 71
5.17.2 Member Enumeration Documentation . . . 71
5.17.2.1 PreprocessState . . . 71
5.17.3 Constructor & Destructor Documentation . . . 71
5.17.3.1 AstNodeIndex() . . . 72
5.17.4 Member Function Documentation . . . 72
5.17.4.1 compile() . . . 72
5.17.4.2 compilePreprocess() . . . 73
5.17.4.3 dump() . . . 73
5.17.4.4 getCollection() . . . 73
5.17.4.5 getIndex() . . . 74
5.18 Tang::AstNodeInteger Class Reference . . . 74
5.18.1 Detailed Description . . . 75
5.18.2 Member Enumeration Documentation . . . 75
5.18.2.1 PreprocessState . . . 75
5.18.3 Constructor & Destructor Documentation . . . 75
5.18.3.1 AstNodeInteger() . . . 75
5.18.4 Member Function Documentation . . . 76
5.18.4.1 compile() . . . 76
5.18.4.2 compilePreprocess() . . . 76
5.18.4.3 dump() . . . 77
5.19 Tang::AstNodePrint Class Reference . . . 77
5.19.1 Detailed Description . . . 78
5.19.2 Member Enumeration Documentation . . . 78
5.19.2.1 PreprocessState . . . 78
5.19.2.2 Type . . . 79
5.19.3 Constructor & Destructor Documentation . . . 79
5.19.3.1 AstNodePrint() . . . 79
5.19.4 Member Function Documentation . . . 79
5.19.4.1 compile() . . . 79
5.19.4.2 compilePreprocess() . . . 80
5.19.4.3 dump() . . . 80
5.20 Tang::AstNodeReturn Class Reference . . . 81

5.20.1 Detailed Description . . . 82
5.20.2 Member Enumeration Documentation . . . 82
5.20.2.1 PreprocessState . . . 82
5.20.3 Constructor & Destructor Documentation . . . 82
5.20.3.1 AstNodeReturn() . . . 82
5.20.4 Member Function Documentation . . . 83
5.20.4.1 compile() . . . 83
5.20.4.2 compilePreprocess() . . . 83
5.20.4.3 dump() . . . 84
5.21 Tang::AstNodeString Class Reference . . . 84
5.21.1 Detailed Description . . . 85
5.21.2 Member Enumeration Documentation . . . 85
5.21.2.1 PreprocessState . . . 85
5.21.3 Constructor & Destructor Documentation . . . 86
5.21.3.1 AstNodeString() . . . 86
5.21.4 Member Function Documentation . . . 86
5.21.4.1 compile() . . . 86
5.21.4.2 compileLiteral() . . . 87
5.21.4.3 compilePreprocess() . . . 87
5.21.4.4 dump() . . . 88
5.22 Tang::AstNodeTernary Class Reference . . . 88
5.22.1 Detailed Description . . . 90
5.22.2 Member Enumeration Documentation . . . 90
5.22.2.1 PreprocessState . . . 90
5.22.3 Constructor & Destructor Documentation . . . 90
5.22.3.1 AstNodeTernary() . . . 90
5.22.4 Member Function Documentation . . . 90
5.22.4.1 compile() . . . 91
5.22.4.2 compilePreprocess() . . . 91
5.22.4.3 dump() . . . 91
5.23 Tang::AstNodeUnary Class Reference . . . 92
5.23.1 Detailed Description . . . 93
5.23.2 Member Enumeration Documentation . . . 93
5.23.2.1 Operator . . . 93
5.23.2.2 PreprocessState . . . 93
5.23.3 Constructor & Destructor Documentation . . . 94
5.23.3.1 AstNodeUnary() . . . 94
5.23.4 Member Function Documentation . . . 94
5.23.4.1 compile() . . . 94
5.23.4.2 compilePreprocess() . . . 95
5.23.4.3 dump() . . . 95
5.24 Tang::AstNodeWhile Class Reference . . . 96

5.24.1 Detailed Description . . . 97
5.24.2 Member Enumeration Documentation . . . 97
5.24.2.1 PreprocessState . . . 97
5.24.3 Constructor & Destructor Documentation . . . 97
5.24.3.1 AstNodeWhile() . . . 97
5.24.4 Member Function Documentation . . . 97
5.24.4.1 compile() . . . 98
5.24.4.2 compilePreprocess() . . . 98
5.24.4.3 dump() . . . 99
5.25 Tang::ComputedExpression Class Reference . . . 99
5.25.1 Detailed Description . . . 101
5.25.2 Member Function Documentation . . . 101
5.25.2.1 __add() . . . 101
5.25.2.2 __assign_index() . . . 102
5.25.2.3 __boolean() . . . 102
5.25.2.4 __divide() . . . 102
5.25.2.5 __equal() . . . 103
5.25.2.6 __float() . . . 103
5.25.2.7 __index() . . . 103
5.25.2.8 __integer() . . . 104
5.25.2.9 __lessThan() . . . 104
5.25.2.10 __modulo() . . . 105
5.25.2.11 __multiply() . . . 105
5.25.2.12 __negative() . . . 105
5.25.2.13 __not() . . . 106
5.25.2.14 __string() . . . 106
5.25.2.15 __subtract() . . . 106
5.25.2.16 dump() . . . 107
5.25.2.17 is_equal() [1/6] . . . 107
5.25.2.18 is_equal() [2/6] . . . 107
5.25.2.19 is_equal() [3/6] . . . 108
5.25.2.20 is_equal() [4/6] . . . 108
5.25.2.21 is_equal() [5/6] . . . 108
5.25.2.22 is_equal() [6/6] . . . 109
5.25.2.23 isCopyNeeded() . . . 109
5.25.2.24 makeCopy() . . . 110
5.26 Tang::ComputedExpressionArray Class Reference . . . 110
5.26.1 Detailed Description . . . 112
5.26.2 Constructor & Destructor Documentation . . . 112
5.26.2.1 ComputedExpressionArray() . . . 112
5.26.3 Member Function Documentation . . . 112
5.26.3.1 __add() . . . 112

5.26.3.2 __assign_index() . . . 113
5.26.3.3 __boolean() . . . 113
5.26.3.4 __divide() . . . 113
5.26.3.5 __equal() . . . 114
5.26.3.6 __float() . . . 114
5.26.3.7 __index() . . . 114
5.26.3.8 __integer() . . . 115
5.26.3.9 __lessThan() . . . 115
5.26.3.10 __modulo() . . . 116
5.26.3.11 __multiply() . . . 116
5.26.3.12 __negative() . . . 116
5.26.3.13 __not() . . . 117
5.26.3.14 __string() . . . 117
5.26.3.15 __subtract() . . . 117
5.26.3.16 dump() . . . 118
5.26.3.17 is_equal() [1/6] . . . 118
5.26.3.18 is_equal() [2/6] . . . 118
5.26.3.19 is_equal() [3/6] . . . 119
5.26.3.20 is_equal() [4/6] . . . 119
5.26.3.21 is_equal() [5/6] . . . 119
5.26.3.22 is_equal() [6/6] . . . 120
5.26.3.23 isCopyNeeded() . . . 120
5.26.3.24 makeCopy() . . . 121
5.27 Tang::ComputedExpressionBoolean Class Reference . . . 121
5.27.1 Detailed Description . . . 123
5.27.2 Constructor & Destructor Documentation . . . 123
5.27.2.1 ComputedExpressionBoolean() . . . 123
5.27.3 Member Function Documentation . . . 123
5.27.3.1 __add() . . . 123
5.27.3.2 __assign_index() . . . 124
5.27.3.3 __boolean() . . . 124
5.27.3.4 __divide() . . . 124
5.27.3.5 __equal() . . . 125
5.27.3.6 __float() . . . 125
5.27.3.7 __index() . . . 125
5.27.3.8 __integer() . . . 126
5.27.3.9 __lessThan() . . . 126
5.27.3.10 __modulo() . . . 126
5.27.3.11 __multiply() . . . 127
5.27.3.12 __negative() . . . 127
5.27.3.13 __not() . . . 128
5.27.3.14 __string() . . . 128

5.27.3.15 __subtract() . . . 128
5.27.3.16 dump() . . . 129
5.27.3.17 is_equal() [1/6] . . . 129
5.27.3.18 is_equal() [2/6] . . . 129
5.27.3.19 is_equal() [3/6] . . . 130
5.27.3.20 is_equal() [4/6] . . . 130
5.27.3.21 is_equal() [5/6] . . . 130
5.27.3.22 is_equal() [6/6] . . . 131
5.27.3.23 isCopyNeeded() . . . 131
5.27.3.24 makeCopy() . . . 131
5.28 Tang::ComputedExpressionCompiledFunction Class Reference . . . 132
5.28.1 Detailed Description . . . 133
5.28.2 Constructor & Destructor Documentation . . . 134
5.28.2.1 ComputedExpressionCompiledFunction() . . . 134
5.28.3 Member Function Documentation . . . 134
5.28.3.1 __add() . . . 134
5.28.3.2 __assign_index() . . . 134
5.28.3.3 __boolean() . . . 135
5.28.3.4 __divide() . . . 135
5.28.3.5 __equal() . . . 136
5.28.3.6 __float() . . . 136
5.28.3.7 __index() . . . 136
5.28.3.8 __integer() . . . 137
5.28.3.9 __lessThan() . . . 137
5.28.3.10 __modulo() . . . 137
5.28.3.11 __multiply() . . . 138
5.28.3.12 __negative() . . . 138
5.28.3.13 __not() . . . 138
5.28.3.14 __string() . . . 139
5.28.3.15 __subtract() . . . 139
5.28.3.16 dump() . . . 139
5.28.3.17 is_equal() [1/6] . . . 139
5.28.3.18 is_equal() [2/6] . . . 140
5.28.3.19 is_equal() [3/6] . . . 140
5.28.3.20 is_equal() [4/6] . . . 141
5.28.3.21 is_equal() [5/6] . . . 141
5.28.3.22 is_equal() [6/6] . . . 141
5.28.3.23 isCopyNeeded() . . . 142
5.28.3.24 makeCopy() . . . 142
5.29 Tang::ComputedExpressionError Class Reference . . . 143
5.29.1 Detailed Description . . . 144
5.29.2 Constructor & Destructor Documentation . . . 144

5.29.2.1 ComputedExpressionError() . . . 145
5.29.3 Member Function Documentation . . . 145
5.29.3.1 __add() . . . 145
5.29.3.2 __assign_index() . . . 145
5.29.3.3 __boolean() . . . 146
5.29.3.4 __divide() . . . 146
5.29.3.5 __equal() . . . 146
5.29.3.6 __float() . . . 147
5.29.3.7 __index() . . . 147
5.29.3.8 __integer() . . . 147
5.29.3.9 __lessThan() . . . 148
5.29.3.10 __modulo() . . . 148
5.29.3.11 __multiply() . . . 148
5.29.3.12 __negative() . . . 149
5.29.3.13 __not() . . . 149
5.29.3.14 __string() . . . 149
5.29.3.15 __subtract() . . . 149
5.29.3.16 dump() . . . 150
5.29.3.17 is_equal() [1/6] . . . 150
5.29.3.18 is_equal() [2/6] . . . 151
5.29.3.19 is_equal() [3/6] . . . 152
5.29.3.20 is_equal() [4/6] . . . 152
5.29.3.21 is_equal() [5/6] . . . 153
5.29.3.22 is_equal() [6/6] . . . 153
5.29.3.23 isCopyNeeded() . . . 153
5.29.3.24 makeCopy() . . . 154
5.30 Tang::ComputedExpressionFloat Class Reference . . . 154
5.30.1 Detailed Description . . . 156
5.30.2 Constructor & Destructor Documentation . . . 156
5.30.2.1 ComputedExpressionFloat() . . . 156
5.30.3 Member Function Documentation . . . 156
5.30.3.1 __add() . . . 156
5.30.3.2 __assign_index() . . . 157
5.30.3.3 __boolean() . . . 157
5.30.3.4 __divide() . . . 157
5.30.3.5 __equal() . . . 158
5.30.3.6 __float() . . . 158
5.30.3.7 __index() . . . 158
5.30.3.8 __integer() . . . 159
5.30.3.9 __lessThan() . . . 159
5.30.3.10 __modulo() . . . 159
5.30.3.11 __multiply() . . . 160

5.30.3.12 __negative() . . . 160
5.30.3.13 __not() . . . 161
5.30.3.14 __string() . . . 161
5.30.3.15 __subtract() . . . 161
5.30.3.16 dump() . . . 162
5.30.3.17 is_equal() [1/6] . . . 162
5.30.3.18 is_equal() [2/6] . . . 162
5.30.3.19 is_equal() [3/6] . . . 163
5.30.3.20 is_equal() [4/6] . . . 163
5.30.3.21 is_equal() [5/6] . . . 164
5.30.3.22 is_equal() [6/6] . . . 164
5.30.3.23 isCopyNeeded() . . . 164
5.30.3.24 makeCopy() . . . 165
5.31 Tang::ComputedExpressionInteger Class Reference . . . 165
5.31.1 Detailed Description . . . 167
5.31.2 Constructor & Destructor Documentation . . . 167
5.31.2.1 ComputedExpressionInteger() . . . 167
5.31.3 Member Function Documentation . . . 167
5.31.3.1 __add() . . . 167
5.31.3.2 __assign_index() . . . 168
5.31.3.3 __boolean() . . . 168
5.31.3.4 __divide() . . . 168
5.31.3.5 __equal() . . . 169
5.31.3.6 __float() . . . 169
5.31.3.7 __index() . . . 169
5.31.3.8 __integer() . . . 170
5.31.3.9 __lessThan() . . . 170
5.31.3.10 __modulo() . . . 170
5.31.3.11 __multiply() . . . 171
5.31.3.12 __negative() . . . 171
5.31.3.13 __not() . . . 172
5.31.3.14 __string() . . . 172
5.31.3.15 __subtract() . . . 172
5.31.3.16 dump() . . . 173
5.31.3.17 is_equal() [1/6] . . . 173
5.31.3.18 is_equal() [2/6] . . . 173
5.31.3.19 is_equal() [3/6] . . . 174
5.31.3.20 is_equal() [4/6] . . . 174
5.31.3.21 is_equal() [5/6] . . . 175
5.31.3.22 is_equal() [6/6] . . . 175
5.31.3.23 isCopyNeeded() . . . 175
5.31.3.24 makeCopy() . . . 176

5.32 Tang::ComputedExpressionString Class Reference . . . 176
5.32.1 Detailed Description . . . 178
5.32.2 Constructor & Destructor Documentation . . . 178
5.32.2.1 ComputedExpressionString() . . . 178
5.32.3 Member Function Documentation . . . 178
5.32.3.1 __add() . . . 178
5.32.3.2 __assign_index() . . . 178
5.32.3.3 __boolean() . . . 180
5.32.3.4 __divide() . . . 180
5.32.3.5 __equal() . . . 181
5.32.3.6 __float() . . . 181
5.32.3.7 __index() . . . 181
5.32.3.8 __integer() . . . 182
5.32.3.9 __lessThan() . . . 182
5.32.3.10 __modulo() . . . 182
5.32.3.11 __multiply() . . . 183
5.32.3.12 __negative() . . . 183
5.32.3.13 __not() . . . 183
5.32.3.14 __string() . . . 184
5.32.3.15 __subtract() . . . 184
5.32.3.16 dump() . . . 184
5.32.3.17 is_equal() [1/6] . . . 184
5.32.3.18 is_equal() [2/6] . . . 185
5.32.3.19 is_equal() [3/6] . . . 185
5.32.3.20 is_equal() [4/6] . . . 186
5.32.3.21 is_equal() [5/6] . . . 186
5.32.3.22 is_equal() [6/6] . . . 186
5.32.3.23 isCopyNeeded() . . . 187
5.32.3.24 makeCopy() . . . 187
5.33 Tang::Error Class Reference . . . 188
5.33.1 Detailed Description . . . 189
5.33.2 Constructor & Destructor Documentation . . . 189
5.33.2.1 Error() [1/2] . . . 189
5.33.2.2 Error() [2/2] . . . 189
5.33.3 Friends And Related Function Documentation . . . 189
5.33.3.1 operator<< . . . 190
5.34 Tang::GarbageCollected Class Reference . . . 190
5.34.1 Detailed Description . . . 192
5.34.2 Constructor & Destructor Documentation . . . 192
5.34.2.1 GarbageCollected() [1/3] . . . 192
5.34.2.2 GarbageCollected() [2/3] . . . 193
5.34.2.3 ~GarbageCollected() . . . 193

5.34.2.4 GarbageCollected() [3/3] . . . 193
5.34.3 Member Function Documentation . . . 193
5.34.3.1 isCopyNeeded() . . . 193
5.34.3.2 make() . . . 194
5.34.3.3 makeCopy() . . . 194
5.34.3.4 operator"!() . . . 195
5.34.3.5 operator"!=() . . . 195
5.34.3.6 operator%() . . . 196
5.34.3.7 operator∗() [1/2] . . . 197
5.34.3.8 operator∗() [2/2] . . . 197
5.34.3.9 operator+() . . . 197
5.34.3.10 operator-() [1/2] . . . 198
5.34.3.11 operator-() [2/2] . . . 198
5.34.3.12 operator->() . . . 199
5.34.3.13 operator/() . . . 199
5.34.3.14 operator<() . . . 200
5.34.3.15 operator<=() . . . 200
5.34.3.16 operator=() [1/2] . . . 201
5.34.3.17 operator=() [2/2] . . . 201
5.34.3.18 operator==() [1/8] . . . 201
5.34.3.19 operator==() [2/8] . . . 202
5.34.3.20 operator==() [3/8] . . . 202
5.34.3.21 operator==() [4/8] . . . 202
5.34.3.22 operator==() [5/8] . . . 203
5.34.3.23 operator==() [6/8] . . . 203
5.34.3.24 operator==() [7/8] . . . 204
5.34.3.25 operator==() [8/8] . . . 204
5.34.3.26 operator>() . . . 205
5.34.3.27 operator>=() . . . 205
5.34.4 Friends And Related Function Documentation . . . 205
5.34.4.1 operator<< . . . 206
5.35 Tang::location Class Reference . . . 206
5.35.1 Detailed Description . . . 207
5.36 Tang::position Class Reference . . . 208
5.36.1 Detailed Description . . . 209
5.37 Tang::Program Class Reference . . . 209
5.37.1 Detailed Description . . . 211
5.37.2 Member Enumeration Documentation . . . 211
5.37.2.1 CodeType . . . 211
5.37.3 Constructor & Destructor Documentation . . . 211
5.37.3.1 Program() . . . 211
5.37.4 Member Function Documentation . . . 212

5.37.4.1 addBreak() . . . 212
5.37.4.2 addBytecode() . . . 212
5.37.4.3 addContinue() . . . 212
5.37.4.4 addIdentifier() . . . 213
5.37.4.5 addIdentifierAssigned() . . . 213
5.37.4.6 addString() . . . 213
5.37.4.7 dumpBytecode() . . . 213
5.37.4.8 execute() . . . 214
5.37.4.9 getAst() . . . 214
5.37.4.10 getBytecode() . . . 214
5.37.4.11 getCode() . . . 215
5.37.4.12 getIdentifiers() . . . 215
5.37.4.13 getIdentifiersAssigned() . . . 215
5.37.4.14 getResult() . . . 215
5.37.4.15 getStrings() . . . 216
5.37.4.16 popBreakStack() . . . 216
5.37.4.17 popContinueStack() . . . 216
5.37.4.18 pushEnvironment() . . . 217
5.37.4.19 setFunctionStackDeclaration() . . . 217
5.37.4.20 setJumpTarget() . . . 218
5.37.5 Member Data Documentation . . . 218
5.37.5.1 functionsDeclared . . . 218
5.38 Tang::SingletonObjectPool< T > Class Template Reference . . . 218
5.38.1 Detailed Description . . . 219
5.38.2 Member Function Documentation . . . 219
5.38.2.1 get() . . . 219
5.38.2.2 getInstance() . . . 219
5.38.2.3 recycle() . . . 219
5.39 Tang::TangBase Class Reference . . . 220
5.39.1 Detailed Description . . . 220
5.39.2 Constructor & Destructor Documentation . . . 220
5.39.2.1 TangBase() . . . 220
5.39.3 Member Function Documentation . . . 220
5.39.3.1 compileScript() . . . 220
5.40 Tang::TangScanner Class Reference . . . 221
5.40.1 Detailed Description . . . 222
5.40.2 Constructor & Destructor Documentation . . . 222
5.40.2.1 TangScanner() . . . 222
5.40.3 Member Function Documentation . . . 222
5.40.3.1 get_next_token() . . . 222

6 File Documentation **225**

6.1 build/generated/location.hh File Reference . . . 225
6.1.1 Detailed Description . . . 226
6.1.2 Function Documentation . . . 226
6.1.2.1 operator<<() [1/2] . . . 226
6.1.2.2 operator<<() [2/2] . . . 227
6.2 include/astNode.hpp File Reference . . . 227
6.2.1 Detailed Description . . . 228
6.3 include/astNodeArray.hpp File Reference . . . 228
6.3.1 Detailed Description . . . 229
6.4 include/astNodeAssign.hpp File Reference . . . 229
6.4.1 Detailed Description . . . 230
6.5 include/astNodeBinary.hpp File Reference . . . 230
6.5.1 Detailed Description . . . 231
6.6 include/astNodeBlock.hpp File Reference . . . 231
6.6.1 Detailed Description . . . 232
6.7 include/astNodeBoolean.hpp File Reference . . . 232
6.7.1 Detailed Description . . . 233
6.8 include/astNodeBreak.hpp File Reference . . . 233
6.8.1 Detailed Description . . . 234
6.9 include/astNodeCast.hpp File Reference . . . 234
6.9.1 Detailed Description . . . 235
6.10 include/astNodeContinue.hpp File Reference . . . 235
6.10.1 Detailed Description . . . 236
6.11 include/astNodeDoWhile.hpp File Reference . . . 236
6.11.1 Detailed Description . . . 237
6.12 include/astNodeFloat.hpp File Reference . . . 237
6.12.1 Detailed Description . . . 238
6.13 include/astNodeFor.hpp File Reference . . . 238
6.13.1 Detailed Description . . . 239
6.14 include/astNodeFunctionCall.hpp File Reference . . . 239
6.14.1 Detailed Description . . . 240
6.15 include/astNodeFunctionDeclaration.hpp File Reference . . . 240
6.15.1 Detailed Description . . . 241
6.16 include/astNodeIdentifier.hpp File Reference . . . 241
6.16.1 Detailed Description . . . 242
6.17 include/astNodeIfElse.hpp File Reference . . . 242
6.17.1 Detailed Description . . . 243
6.18 include/astNodeIndex.hpp File Reference . . . 243
6.18.1 Detailed Description . . . 244
6.19 include/astNodeInteger.hpp File Reference . . . 244
6.19.1 Detailed Description . . . 245
6.20 include/astNodePrint.hpp File Reference . . . 245

6.20.1 Detailed Description . . . 246
6.21 include/astNodeReturn.hpp File Reference . . . 246
6.21.1 Detailed Description . . . 247
6.22 include/astNodeString.hpp File Reference . . . 247
6.22.1 Detailed Description . . . 248
6.23 include/astNodeTernary.hpp File Reference . . . 248
6.23.1 Detailed Description . . . 249
6.24 include/astNodeUnary.hpp File Reference . . . 249
6.24.1 Detailed Description . . . 250
6.25 include/astNodeWhile.hpp File Reference . . . 250
6.25.1 Detailed Description . . . 251
6.26 include/computedExpression.hpp File Reference . . . 251
6.26.1 Detailed Description . . . 251
6.27 include/computedExpressionArray.hpp File Reference . . . 252
6.27.1 Detailed Description . . . 252
6.28 include/computedExpressionBoolean.hpp File Reference . . . 253
6.28.1 Detailed Description . . . 253
6.29 include/computedExpressionCompiledFunction.hpp File Reference . . . 254
6.29.1 Detailed Description . . . 254
6.30 include/computedExpressionError.hpp File Reference . . . 255
6.30.1 Detailed Description . . . 255
6.31 include/computedExpressionFloat.hpp File Reference . . . 256
6.31.1 Detailed Description . . . 256
6.32 include/computedExpressionInteger.hpp File Reference . . . 257
6.32.1 Detailed Description . . . 257
6.33 include/computedExpressionString.hpp File Reference . . . 258
6.33.1 Detailed Description . . . 258
6.34 include/error.hpp File Reference . . . 259
6.34.1 Detailed Description . . . 259
6.35 include/garbageCollected.hpp File Reference . . . 260
6.35.1 Detailed Description . . . 260
6.36 include/macros.hpp File Reference . . . 260
6.36.1 Detailed Description . . . 261
6.37 include/opcode.hpp File Reference . . . 261
6.37.1 Detailed Description . . . 261
6.37.2 Enumeration Type Documentation . . . 261
6.37.2.1 Opcode . . . 261
6.38 include/program.hpp File Reference . . . 262
6.38.1 Detailed Description . . . 263
6.39 include/singletonObjectPool.hpp File Reference . . . 264
6.39.1 Detailed Description . . . 264
6.40 include/tang.hpp File Reference . . . 265

6.40.1 Detailed Description . . . 265
6.41 include/tangBase.hpp File Reference . . . 266
6.41.1 Detailed Description . . . 266
6.42 include/tangScanner.hpp File Reference . . . 267
6.42.1 Detailed Description . . . 268
6.43 src/astNode.cpp File Reference . . . 268
6.43.1 Detailed Description . . . 268
6.44 src/astNodeArray.cpp File Reference . . . 268
6.44.1 Detailed Description . . . 269
6.45 src/astNodeAssign.cpp File Reference . . . 269
6.45.1 Detailed Description . . . 270
6.46 src/astNodeBinary.cpp File Reference . . . 270
6.46.1 Detailed Description . . . 271
6.47 src/astNodeBlock.cpp File Reference . . . 271
6.47.1 Detailed Description . . . 271
6.48 src/astNodeBoolean.cpp File Reference . . . 271
6.48.1 Detailed Description . . . 272
6.49 src/astNodeBreak.cpp File Reference . . . 272
6.49.1 Detailed Description . . . 273
6.50 src/astNodeCast.cpp File Reference . . . 273
6.50.1 Detailed Description . . . 273
6.51 src/astNodeContinue.cpp File Reference . . . 273
6.51.1 Detailed Description . . . 274
6.52 src/astNodeDoWhile.cpp File Reference . . . 274
6.52.1 Detailed Description . . . 275
6.53 src/astNodeFloat.cpp File Reference . . . 275
6.53.1 Detailed Description . . . 276
6.54 src/astNodeFor.cpp File Reference . . . 276
6.54.1 Detailed Description . . . 276
6.55 src/astNodeFunctionCall.cpp File Reference . . . 276
6.55.1 Detailed Description . . . 277
6.56 src/astNodeFunctionDeclaration.cpp File Reference . . . 277
6.56.1 Detailed Description . . . 278
6.57 src/astNodeIdentifier.cpp File Reference . . . 278
6.57.1 Detailed Description . . . 279
6.58 src/astNodeIfElse.cpp File Reference . . . 279
6.58.1 Detailed Description . . . 279
6.59 src/astNodeIndex.cpp File Reference . . . 279
6.59.1 Detailed Description . . . 280
6.60 src/astNodeInteger.cpp File Reference . . . 280
6.60.1 Detailed Description . . . 281
6.61 src/astNodePrint.cpp File Reference . . . 281

6.61.1 Detailed Description . . . 281
6.62 src/astNodeReturn.cpp File Reference . . . 281
6.62.1 Detailed Description . . . 282
6.63 src/astNodeString.cpp File Reference . . . 282
6.63.1 Detailed Description . . . 283
6.64 src/astNodeTernary.cpp File Reference . . . 283
6.64.1 Detailed Description . . . 284
6.65 src/astNodeUnary.cpp File Reference . . . 284
6.65.1 Detailed Description . . . 284
6.66 src/astNodeWhile.cpp File Reference . . . 284
6.66.1 Detailed Description . . . 285
6.67 src/computedExpression.cpp File Reference . . . 285
6.67.1 Detailed Description . . . 286
6.68 src/computedExpressionArray.cpp File Reference . . . 286
6.68.1 Detailed Description . . . 287
6.69 src/computedExpressionBoolean.cpp File Reference . . . 287
6.69.1 Detailed Description . . . 287
6.70 src/computedExpressionCompiledFunction.cpp File Reference . . . 287
6.70.1 Detailed Description . . . 288
6.71 src/computedExpressionError.cpp File Reference . . . 288
6.71.1 Detailed Description . . . 289
6.72 src/computedExpressionFloat.cpp File Reference . . . 289
6.72.1 Detailed Description . . . 289
6.73 src/computedExpressionInteger.cpp File Reference . . . 289
6.73.1 Detailed Description . . . 290
6.74 src/computedExpressionString.cpp File Reference . . . 290
6.74.1 Detailed Description . . . 291
6.75 src/error.cpp File Reference . . . 291
6.75.1 Detailed Description . . . 291
6.75.2 Function Documentation . . . 291
6.75.2.1 operator<<() . . . 291
6.76 src/program-dumpBytecode.cpp File Reference . . . 292
6.76.1 Detailed Description . . . 292
6.76.2 Macro Definition Documentation . . . 292
6.76.2.1 DUMPPROGRAMCHECK . . . 293
6.77 src/program-execute.cpp File Reference . . . 293
6.77.1 Detailed Description . . . 294
6.77.2 Macro Definition Documentation . . . 294
6.77.2.1 EXECUTEPROGRAMCHECK . . . 294
6.77.2.2 STACKCHECK . . . 294
6.78 src/program.cpp File Reference . . . 294
6.78.1 Detailed Description . . . 295

6.79 src/tangBase.cpp File Reference 295
6.79.1 Detailed Description 296
6.80 test/test.cpp File Reference 296
6.80.1 Detailed Description 297
6.81 test/testGarbageCollected.cpp File Reference 297
6.81.1 Detailed Description 298
6.82 test/testSingletonObjectPool.cpp File Reference 298
6.82.1 Detailed Description 298

Index **299**

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::AstNodeArray . . . 15
Tang::AstNodeAssign . . . 19
Tang::AstNodeBinary . . . 22
Tang::AstNodeBlock . . . 26
Tang::AstNodeBoolean . . . 30
Tang::AstNodeBreak . . . 34
Tang::AstNodeCast . . . 37
Tang::AstNodeContinue . . . 41
Tang::AstNodeDoWhile . . . 44
Tang::AstNodeFloat . . . 48
Tang::AstNodeFor . . . 51
Tang::AstNodeFunctionCall . . . 55
Tang::AstNodeFunctionDeclaration . . . 58
Tang::AstNodeIdentifier . . . 62
Tang::AstNodeIfElse . . . 66
Tang::AstNodeIndex . . . 70
Tang::AstNodeInteger . . . 74
Tang::AstNodePrint . . . 77
Tang::AstNodeReturn . . . 81
Tang::AstNodeString . . . 84
Tang::AstNodeTernary . . . 88
Tang::AstNodeUnary . . . 92
Tang::AstNodeWhile . . . 96
Tang::ComputedExpression . . . 99
Tang::ComputedExpressionArray . . . 110
Tang::ComputedExpressionBoolean . . . 121
Tang::ComputedExpressionCompiledFunction . . . 132
Tang::ComputedExpressionError . . . 143
Tang::ComputedExpressionFloat . . . 154
Tang::ComputedExpressionInteger . . . 165
Tang::ComputedExpressionString . . . 176
Tang::Error . . . 188
Tang::GarbageCollected . . . 190
Tang::location . . . 206

Tang::position 208
Tang::Program 209
Tang::SingletonObjectPool< T > 218
Tang::TangBase 220
TangTangFlexLexer
    Tang::TangScanner 221

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::AstNodeArray
: An AstNode that represents an array literal . . . 15

Tang::AstNodeAssign
: An AstNode that represents a binary expression . . . 19

Tang::AstNodeBinary
: An AstNode that represents a binary expression . . . 22

Tang::AstNodeBlock
: An AstNode that represents a code block . . . 26

Tang::AstNodeBoolean
: An AstNode that represents a boolean literal . . . 30

Tang::AstNodeBreak
: An AstNode that represents a `break` statement . . . 34

Tang::AstNodeCast
: An AstNode that represents a typecast of an expression . . . 37

Tang::AstNodeContinue
: An AstNode that represents a `continue` statement . . . 41

Tang::AstNodeDoWhile
: An AstNode that represents a do..while statement . . . 44

Tang::AstNodeFloat
: An AstNode that represents an float literal . . . 48

Tang::AstNodeFor
: An AstNode that represents an if() statement . . . 51

Tang::AstNodeFunctionCall
: An AstNode that represents a function call . . . 55

Tang::AstNodeFunctionDeclaration
: An AstNode that represents a function declaration . . . 58

Tang::AstNodeIdentifier
: An AstNode that represents an identifier . . . 62

Tang::AstNodeIfElse
: An AstNode that represents an if..else statement . . . 66

Tang::AstNodeIndex
: An AstNode that represents an index into a collection . . . 70

Tang::AstNodeInteger
: An AstNode that represents an integer literal . . . 74

Tang::AstNodePrint
An AstNode that represents a print typeeration 77
Tang::AstNodeReturn
An AstNode that represents a `return` statement 81
Tang::AstNodeString
An AstNode that represents a string literal 84
Tang::AstNodeTernary
An AstNode that represents a ternary expression 88
Tang::AstNodeUnary
An AstNode that represents a unary negation 92
Tang::AstNodeWhile
An AstNode that represents a while statement 96
Tang::ComputedExpression
Represents the result of a computation that has been executed 99
Tang::ComputedExpressionArray
Represents an Array that is the result of a computation 110
Tang::ComputedExpressionBoolean
Represents an Boolean that is the result of a computation 121
Tang::ComputedExpressionCompiledFunction
Represents a Compiled Function declared in the script 132
Tang::ComputedExpressionError
Represents a Runtime Error 143
Tang::ComputedExpressionFloat
Represents a Float that is the result of a computation 154
Tang::ComputedExpressionInteger
Represents an Integer that is the result of a computation 165
Tang::ComputedExpressionString
Represents a String that is the result of a computation 176
Tang::Error
Used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error 188
Tang::GarbageCollected
A container that acts as a resource-counting garbage collector for the specified type 190
Tang::location
Two points in a source file 206
Tang::position
A point in a source file 208
Tang::Program
Represents a compiled script or template that may be executed 209
Tang::SingletonObjectPool< T >
A thread-safe, singleton object pool of the designated type 218
Tang::TangBase
The base class for the Tang programming language 220
Tang::TangScanner
The Flex lexer class for the main Tang language 221

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 . . . 225
include/astNode.hpp
Declare the Tang::AstNode base class . . . 227
include/astNodeArray.hpp
Declare the Tang::AstNodeArray class . . . 228
include/astNodeAssign.hpp
Declare the Tang::AstNodeAssign class . . . 229
include/astNodeBinary.hpp
Declare the Tang::AstNodeBinary class . . . 230
include/astNodeBlock.hpp
Declare the Tang::AstNodeBlock class . . . 231
include/astNodeBoolean.hpp
Declare the Tang::AstNodeBoolean class . . . 232
include/astNodeBreak.hpp
Declare the Tang::AstNodeBreak class . . . 233
include/astNodeCast.hpp
Declare the Tang::AstNodeCast class . . . 234
include/astNodeContinue.hpp
Declare the Tang::AstNodeContinue class . . . 235
include/astNodeDoWhile.hpp
Declare the Tang::AstNodeDoWhile class . . . 236
include/astNodeFloat.hpp
Declare the Tang::AstNodeFloat class . . . 237
include/astNodeFor.hpp
Declare the Tang::AstNodeFor class . . . 238
include/astNodeFunctionCall.hpp
Declare the Tang::AstNodeFunctionCall class . . . 239
include/astNodeFunctionDeclaration.hpp
Declare the Tang::AstNodeFunctionDeclaration class . . . 240
include/astNodeIdentifier.hpp
Declare the Tang::AstNodeIdentifier class . . . 241
include/astNodeIfElse.hpp
Declare the Tang::AstNodeIfElse class . . . 242
include/astNodeIndex.hpp
Declare the Tang::AstNodeIndex class . . . 243

include/astNodeInteger.hpp
Declare the Tang::AstNodeInteger class . . . 244
include/astNodePrint.hpp
Declare the Tang::AstNodePrint class . . . 245
include/astNodeReturn.hpp
Declare the Tang::AstNodeReturn class . . . 246
include/astNodeString.hpp
Declare the Tang::AstNodeString class . . . 247
include/astNodeTernary.hpp
Declare the Tang::AstNodeTernary class . . . 248
include/astNodeUnary.hpp
Declare the Tang::AstNodeUnary class . . . 249
include/astNodeWhile.hpp
Declare the Tang::AstNodeWhile class . . . 250
include/computedExpression.hpp
Declare the Tang::ComputedExpression base class . . . 251
include/computedExpressionArray.hpp
Declare the Tang::ComputedExpressionArray class . . . 252
include/computedExpressionBoolean.hpp
Declare the Tang::ComputedExpressionBoolean class . . . 253
include/computedExpressionCompiledFunction.hpp
Declare the Tang::ComputedExpressionCompiledFunction class . . . 254
include/computedExpressionError.hpp
Declare the Tang::ComputedExpressionError class . . . 255
include/computedExpressionFloat.hpp
Declare the Tang::ComputedExpressionFloat class . . . 256
include/computedExpressionInteger.hpp
Declare the Tang::ComputedExpressionInteger class . . . 257
include/computedExpressionString.hpp
Declare the Tang::ComputedExpressionString class . . . 258
include/error.hpp
Declare the Tang::Error class used to describe syntax and runtime errors . . . 259
include/garbageCollected.hpp
Declare the Tang::GarbageCollected class . . . 260
include/macros.hpp
Contains generic macros . . . 260
include/opcode.hpp
Declare the Opcodes used in the Bytecode representation of a program . . . 261
include/program.hpp
Declare the Tang::Program class used to compile and execute source code . . . 262
include/singletonObjectPool.hpp
Declare the Tang::SingletonObjectPool class . . . 264
include/tang.hpp
Header file supplied for use by 3rd party code so that they can easily include all necessary headers . . . 265
include/tangBase.hpp
Declare the Tang::TangBase class used to interact with Tang . . . 266
include/tangScanner.hpp
Declare the Tang::TangScanner used to tokenize a Tang script . . . 267
src/astNode.cpp
Define the Tang::AstNode class . . . 268
src/astNodeArray.cpp
Define the Tang::AstNodeArray class . . . 268
src/astNodeAssign.cpp
Define the Tang::AstNodeAssign class . . . 269
src/astNodeBinary.cpp
Define the Tang::AstNodeBinary class . . . 270

src/astNodeBlock.cpp
Define the Tang::AstNodeBlock class . . . 271
src/astNodeBoolean.cpp
Define the Tang::AstNodeBoolean class . . . 271
src/astNodeBreak.cpp
Define the Tang::AstNodeBreak class . . . 272
src/astNodeCast.cpp
Define the Tang::AstNodeCast class . . . 273
src/astNodeContinue.cpp
Define the Tang::AstNodeContinue class . . . 273
src/astNodeDoWhile.cpp
Define the Tang::AstNodeDoWhile class . . . 274
src/astNodeFloat.cpp
Define the Tang::AstNodeFloat class . . . 275
src/astNodeFor.cpp
Define the Tang::AstNodeFor class . . . 276
src/astNodeFunctionCall.cpp
Define the Tang::AstNodeFunctionCall class . . . 276
src/astNodeFunctionDeclaration.cpp
Define the Tang::AstNodeFunctionDeclaration class . . . 277
src/astNodeIdentifier.cpp
Define the Tang::AstNodeIdentifier class . . . 278
src/astNodeIfElse.cpp
Define the Tang::AstNodeIfElse class . . . 279
src/astNodeIndex.cpp
Define the Tang::AstNodeIndex class . . . 279
src/astNodeInteger.cpp
Define the Tang::AstNodeInteger class . . . 280
src/astNodePrint.cpp
Define the Tang::AstNodePrint class . . . 281
src/astNodeReturn.cpp
Define the Tang::AstNodeReturn class . . . 281
src/astNodeString.cpp
Define the Tang::AstNodeString class . . . 282
src/astNodeTernary.cpp
Define the Tang::AstNodeTernary class . . . 283
src/astNodeUnary.cpp
Define the Tang::AstNodeUnary class . . . 284
src/astNodeWhile.cpp
Define the Tang::AstNodeWhile class . . . 284
src/computedExpression.cpp
Define the Tang::ComputedExpression class . . . 285
src/computedExpressionArray.cpp
Define the Tang::ComputedExpressionArray class . . . 286
src/computedExpressionBoolean.cpp
Define the Tang::ComputedExpressionBoolean class . . . 287
src/computedExpressionCompiledFunction.cpp
Define the Tang::ComputedExpressionCompiledFunction class . . . 287
src/computedExpressionError.cpp
Define the Tang::ComputedExpressionError class . . . 288
src/computedExpressionFloat.cpp
Define the Tang::ComputedExpressionFloat class . . . 289
src/computedExpressionInteger.cpp
Define the Tang::ComputedExpressionInteger class . . . 289
src/computedExpressionString.cpp
Define the Tang::ComputedExpressionString class . . . 290
src/error.cpp
Define the Tang::Error class . . . 291

src/program-dumpBytecode.cpp
Define the Tang::Program::dumpBytecode method 292
src/program-execute.cpp
Define the Tang::Program::execute method 293
src/program.cpp
Define the Tang::Program class 294
src/tangBase.cpp
Define the Tang::TangBase class 295
test/test.cpp
Test the general language behaviors 296
test/testGarbageCollected.cpp
Test the generic behavior of the Tang::GarbageCollected class 297
test/testSingletonObjectPool.cpp
Test the generic behavior of the Tang::SingletonObjectPool class 298

Chapter 5

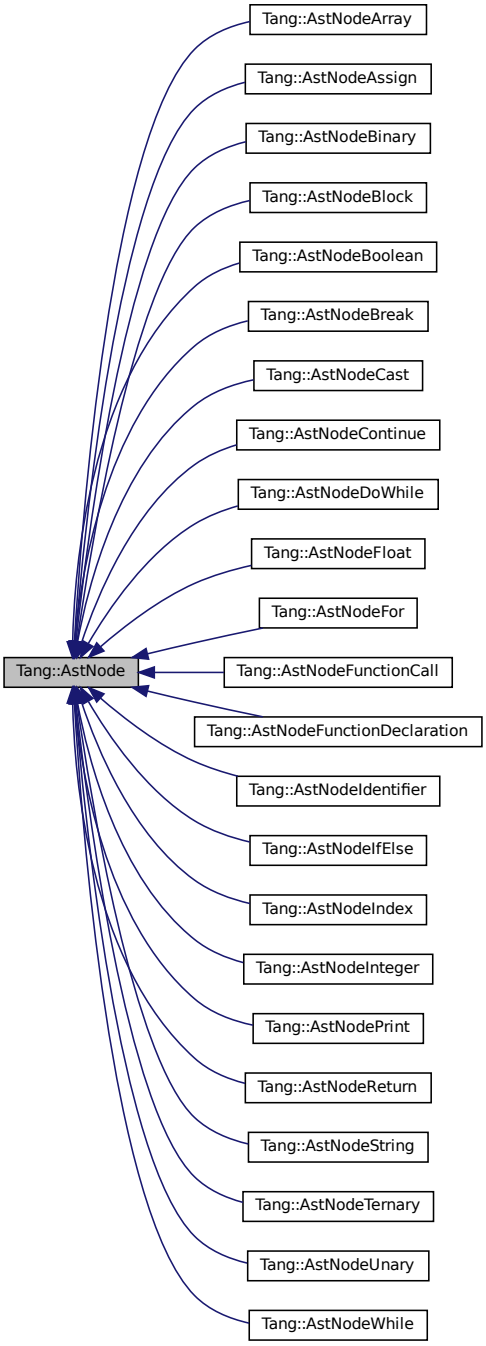
Class Documentation

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 Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNode (Tang::location location)

  The generic constructor.
- virtual ~AstNode ()

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const

  Run any preprocess analysis needed before compilation.

5.1.1 Detailed Description

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

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

5.1.2 Member Enumeration Documentation

5.1.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.1.3 Constructor & Destructor Documentation

5.1.3.1 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.

5.1.4 Member Function Documentation

5.1.4.1 compile()

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

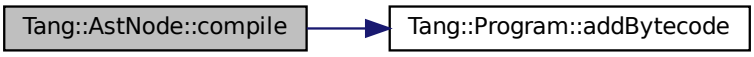
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeReturn, Tang::AstNodePrint, Tang::AstNodeInteger, Tang::AstNodeIndex, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeFloat, Tang::AstNodeDoWhile, Tang::AstNodeContinue, Tang::AstNodeCast, Tang::AstNodeBreak, Tang::AstNodeBoolean, Tang::AstNodeBlock, Tang::AstNodeBinary, Tang::AstNodeAssign, and Tang::AstNodeArray.

Here is the call graph for this function:



5.1.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

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

5.1.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

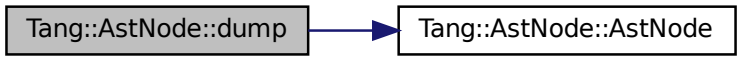
indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeReturn, Tang::AstNodePrint, Tang::AstNodeInteger, Tang::AstNodeIndex, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeFloat, Tang::AstNodeDoWhile, Tang::AstNodeContinue, Tang::AstNodeCast, Tang::AstNodeBreak, Tang::AstNodeBoolean, Tang::AstNodeBlock, Tang::AstNodeBinary, Tang::AstNodeAssign, and Tang::AstNodeArray.

Here is the call graph for this function:



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

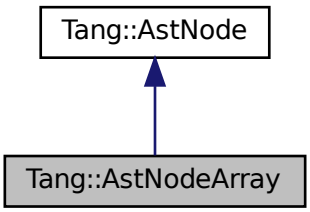
- include/astNode.hpp
- src/astNode.cpp

5.2 Tang::AstNodeArray Class Reference

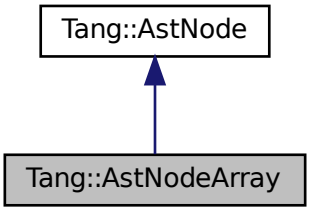
An AstNode that represents an array literal.

```
#include <astNodeArray.hpp>
```

Inheritance diagram for Tang::AstNodeArray:



Collaboration diagram for Tang::AstNodeArray:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeArray (std::vector< std::shared_ptr< Tang::AstNode >> contents, Tang::location location)

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.2.1 Detailed Description

An AstNode that represents an array literal.

5.2.2 Member Enumeration Documentation

5.2.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.2.3 Constructor & Destructor Documentation

5.2.3.1 AstNodeArray()

```
AstNodeArray::AstNodeArray (
            std::vector< std::shared_ptr< Tang::AstNode >> contents,
            Tang::location location )
```

The constructor.

Parameters

contents	The contents of the array.
location	The location associated with the expression.

5.2.4 Member Function Documentation

5.2.4.1 compile()

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

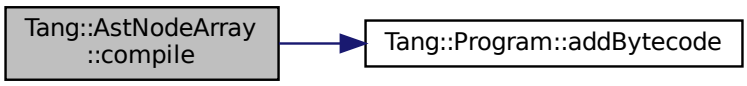
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.2.4.2 compilePreprocess()

```
void AstNodeArray::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.2.4.3 dump()

```
string AstNodeArray::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:

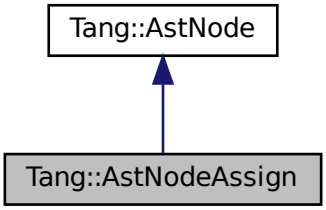
- include/astNodeArray.hpp
- src/astNodeArray.cpp

5.3 Tang::AstNodeAssign Class Reference

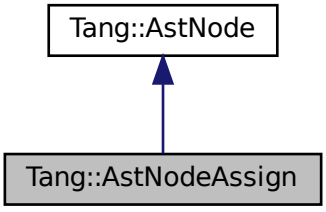
An AstNode that represents a binary expression.

`#include <astNodeAssign.hpp>`

Inheritance diagram for Tang::AstNodeAssign:



Collaboration diagram for Tang::AstNodeAssign:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.3.1 Detailed Description

An AstNode that represents a binary expression.

5.3.2 Member Enumeration Documentation

5.3.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.3.3 Constructor & Destructor Documentation

5.3.3.1 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.

5.3.4 Member Function Documentation

5.3.4.1 compile()

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

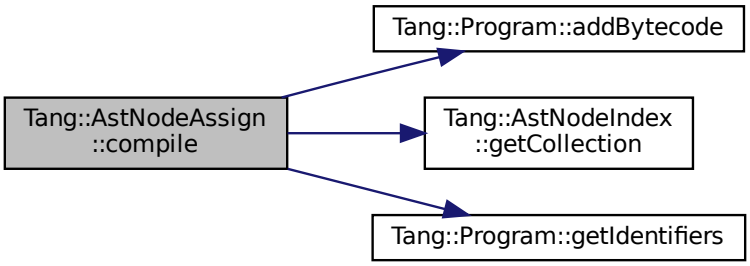
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.3.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.3.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

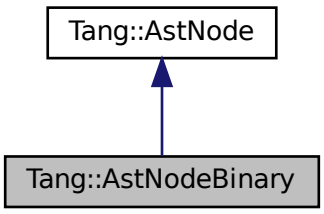
- include/astNodeAssign.hpp
- src/astNodeAssign.cpp

5.4 Tang::AstNodeBinary Class Reference

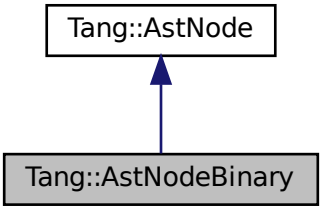
An AstNode that represents a binary expression.

`#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.
- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.4.1 Detailed Description

An AstNode that represents a binary expression.

5.4.2 Member Enumeration Documentation

5.4.2.1 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.

5.4.2.2 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.4.3 Constructor & Destructor Documentation

5.4.3.1 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.

5.4.4 Member Function Documentation

5.4.4.1 compile()

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

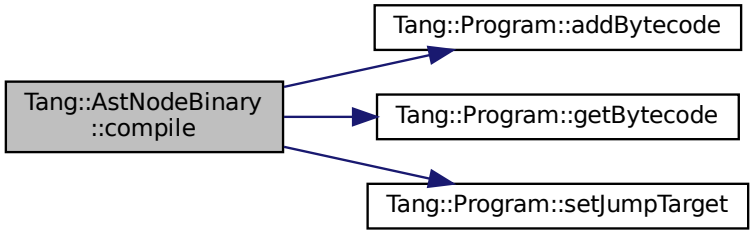
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.4.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.4.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

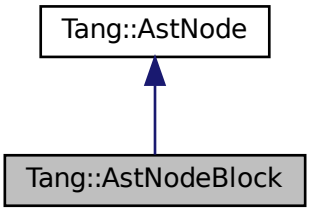
- include/astNodeBinary.hpp
- src/astNodeBinary.cpp

5.5 Tang::AstNodeBlock Class Reference

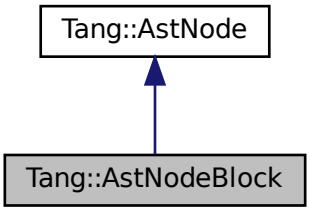
An AstNode that represents a code block.

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

Inheritance diagram for Tang::AstNodeBlock:



Collaboration diagram for Tang::AstNodeBlock:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.5.1 Detailed Description

An AstNode that represents a code block.

5.5.2 Member Enumeration Documentation

5.5.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.5.3 Constructor & Destructor Documentation

5.5.3.1 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.

5.5.4 Member Function Documentation

5.5.4.1 compile()

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

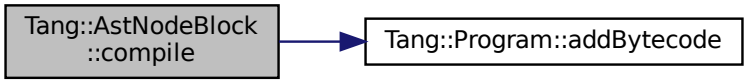
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.5.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.5.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

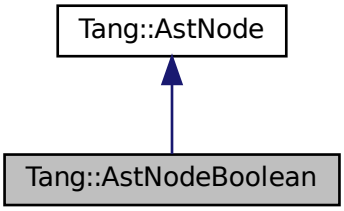
- include/astNodeBlock.hpp
- src/astNodeBlock.cpp

5.6 Tang::AstNodeBoolean Class Reference

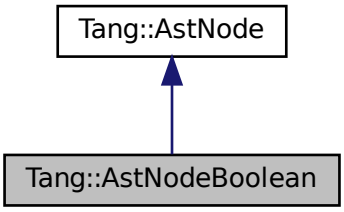
An AstNode that represents a boolean literal.

`#include <astNodeBoolean.hpp>`

Inheritance diagram for Tang::AstNodeBoolean:



Collaboration diagram for Tang::AstNodeBoolean:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const

  Run any preprocess analysis needed before compilation.

5.6.1 Detailed Description

An AstNode that represents a boolean literal.

5.6.2 Member Enumeration Documentation

5.6.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.6.3 Constructor & Destructor Documentation

5.6.3.1 AstNodeBoolean()

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

The constructor.

Parameters

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

5.6.4 Member Function Documentation

5.6.4.1 compile()

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

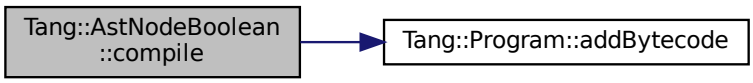
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.6.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

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

5.6.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

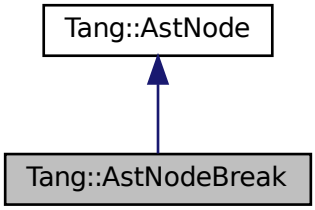
- include/astNodeBoolean.hpp
- src/astNodeBoolean.cpp

5.7 Tang::AstNodeBreak Class Reference

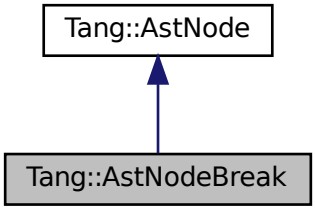
An AstNode that represents a `break` statement.

`#include <astNodeBreak.hpp>`

Inheritance diagram for Tang::AstNodeBreak:



Collaboration diagram for Tang::AstNodeBreak:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeBreak (Tang::location location)

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const

  Run any preprocess analysis needed before compilation.

5.7.1 Detailed Description

An AstNode that represents a `break` statement.

5.7.2 Member Enumeration Documentation

5.7.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.7.3 Constructor & Destructor Documentation

5.7.3.1 AstNodeBreak()

```
AstNodeBreak::AstNodeBreak (
            Tang::location location )
```

The constructor.

Parameters

location	The location associated with the expression.

5.7.4 Member Function Documentation

5.7.4.1 compile()

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

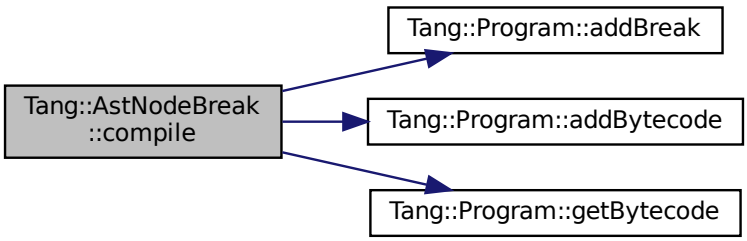
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.7.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString,

Tang::AstNodeReturn, Tang::AstNodePrint, Tang::AstNodeIndex, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBlock, Tang::AstNodeBinary, Tang::AstNodeAssign, and Tang::AstNodeArray.

5.7.4.3 dump()

```
string AstNodeBreak::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:

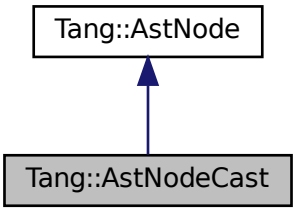
- include/astNodeBreak.hpp
- src/astNodeBreak.cpp

5.8 Tang::AstNodeCast Class Reference

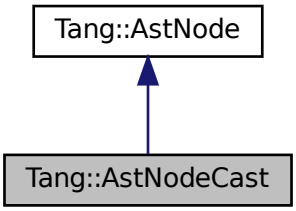
An AstNode that represents a typecast of an expression.

```
#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.
- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.8.1 Detailed Description

An AstNode that represents a typecast of an expression.

5.8.2 Member Enumeration Documentation

5.8.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.8.2.2 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.

5.8.3 Constructor & Destructor Documentation

5.8.3.1 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.

5.8.4 Member Function Documentation

5.8.4.1 compile()

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

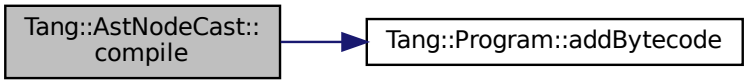
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.8.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.8.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

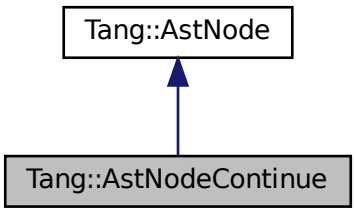
- include/astNodeCast.hpp
- src/astNodeCast.cpp

5.9 Tang::AstNodeContinue Class Reference

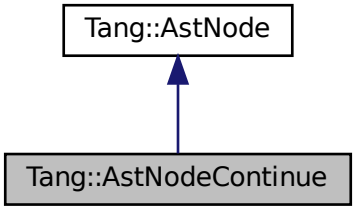
An AstNode that represents a `continue` statement.

`#include <astNodeContinue.hpp>`

Inheritance diagram for Tang::AstNodeContinue:



Collaboration diagram for Tang::AstNodeContinue:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeContinue (Tang::location location)

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const

  Run any preprocess analysis needed before compilation.

5.9.1 Detailed Description

An AstNode that represents a `continue` statement.

5.9.2 Member Enumeration Documentation

5.9.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.9.3 Constructor & Destructor Documentation

5.9.3.1 AstNodeContinue()

```
AstNodeContinue::AstNodeContinue (
            Tang::location location )
```

The constructor.

Parameters

location	The location associated with the expression.

5.9.4 Member Function Documentation

5.9.4.1 compile()

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

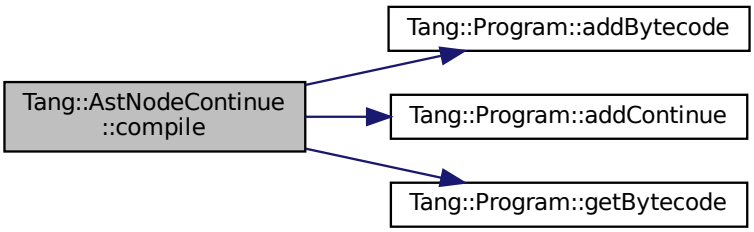
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.9.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString,

Tang::AstNodeReturn, Tang::AstNodePrint, Tang::AstNodeIndex, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBlock, Tang::AstNodeBinary, Tang::AstNodeAssign, and Tang::AstNodeArray.

5.9.4.3 dump()

```
string AstNodeContinue::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:

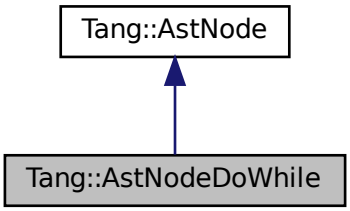
- include/astNodeContinue.hpp
- src/astNodeContinue.cpp

5.10 Tang::AstNodeDoWhile Class Reference

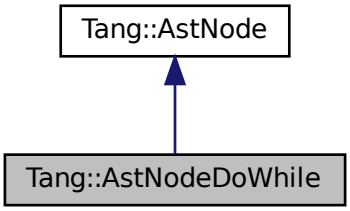
An AstNode that represents a do..while statement.

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

Inheritance diagram for Tang::AstNodeDoWhile:



Collaboration diagram for Tang::AstNodeDoWhile:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.10.1 Detailed Description

An AstNode that represents a do..while statement.

5.10.2 Member Enumeration Documentation

5.10.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.10.3 Constructor & Destructor Documentation

5.10.3.1 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.

5.10.4 Member Function Documentation

5.10.4.1 compile()

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

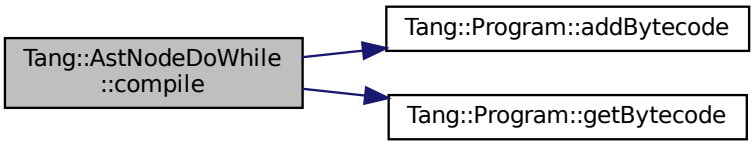
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.10.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.10.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

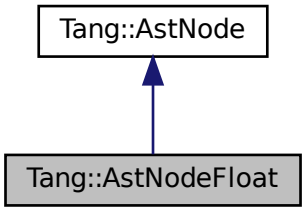
- include/astNodeDoWhile.hpp
- src/astNodeDoWhile.cpp

5.11 Tang::AstNodeFloat Class Reference

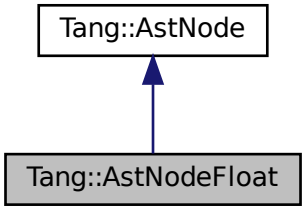
An AstNode that represents an float literal.

`#include <astNodeFloat.hpp>`

Inheritance diagram for Tang::AstNodeFloat:



Collaboration diagram for Tang::AstNodeFloat:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeFloat (Tang::float_t number, Tang::location location)

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const

  Run any preprocess analysis needed before compilation.

5.11.1 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.

5.11.2 Member Enumeration Documentation

5.11.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.11.3 Constructor & Destructor Documentation

5.11.3.1 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.

5.11.4 Member Function Documentation

5.11.4.1 compile()

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

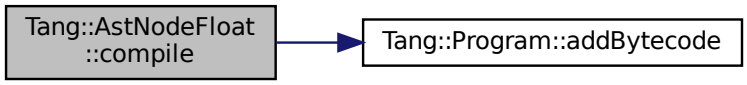
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.11.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

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

5.11.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

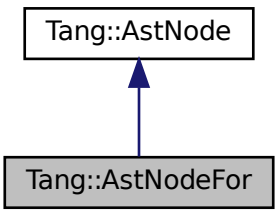
- include/astNodeFloat.hpp
- src/astNodeFloat.cpp

5.12 Tang::AstNodeFor Class Reference

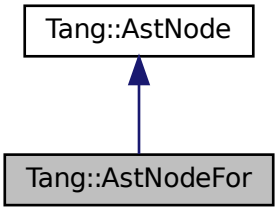
An AstNode that represents an if() statement.

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

Inheritance diagram for Tang::AstNodeFor:



Collaboration diagram for Tang::AstNodeFor:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.12.1 Detailed Description

An AstNode that represents an if() statement.

5.12.2 Member Enumeration Documentation

5.12.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.12.3 Constructor & Destructor Documentation

5.12.3.1 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.

5.12.4 Member Function Documentation

5.12.4.1 compile()

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

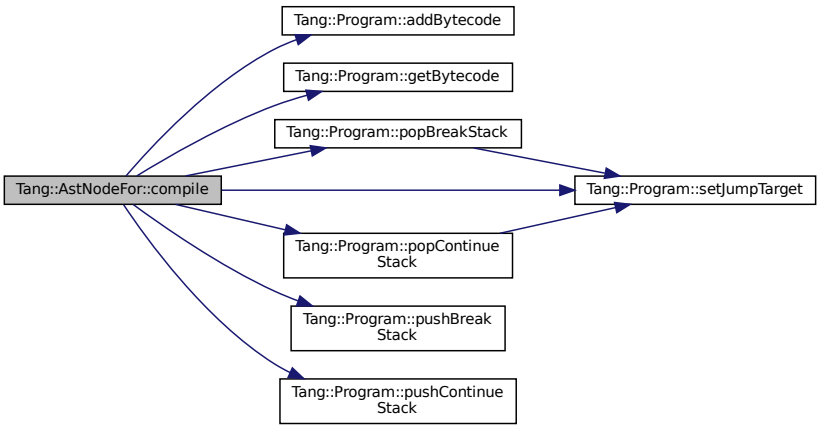
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.12.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.12.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

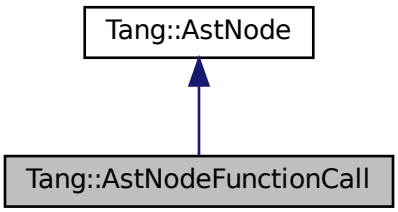
- include/astNodeFor.hpp
- src/astNodeFor.cpp

5.13 Tang::AstNodeFunctionCall Class Reference

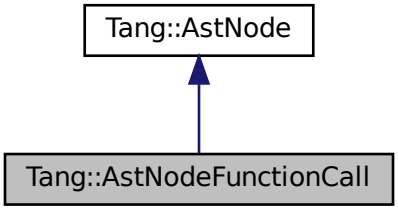
An AstNode that represents a function call.

`#include <astNodeFunctionCall.hpp>`

Inheritance diagram for Tang::AstNodeFunctionCall:



Collaboration diagram for Tang::AstNodeFunctionCall:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.13.1 Detailed Description

An AstNode that represents a function call.

5.13.2 Member Enumeration Documentation

5.13.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.13.3 Constructor & Destructor Documentation

5.13.3.1 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.

5.13.4 Member Function Documentation

5.13.4.1 compile()

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

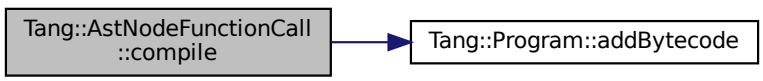
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.13.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.13.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

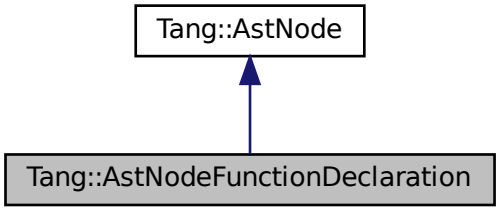
- include/astNodeFunctionCall.hpp
- src/astNodeFunctionCall.cpp

5.14 Tang::AstNodeFunctionDeclaration Class Reference

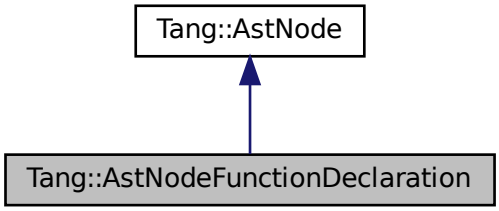
An AstNode that represents a function declaration.

```
#include <astNodeFunctionDeclaration.hpp>
```

Inheritance diagram for Tang::AstNodeFunctionDeclaration:



Collaboration diagram for Tang::AstNodeFunctionDeclaration:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.14.1 Detailed Description

An AstNode that represents a function declaration.

5.14.2 Member Enumeration Documentation

5.14.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.14.3 Constructor & Destructor Documentation

5.14.3.1 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.

5.14.4 Member Function Documentation

5.14.4.1 compile()

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

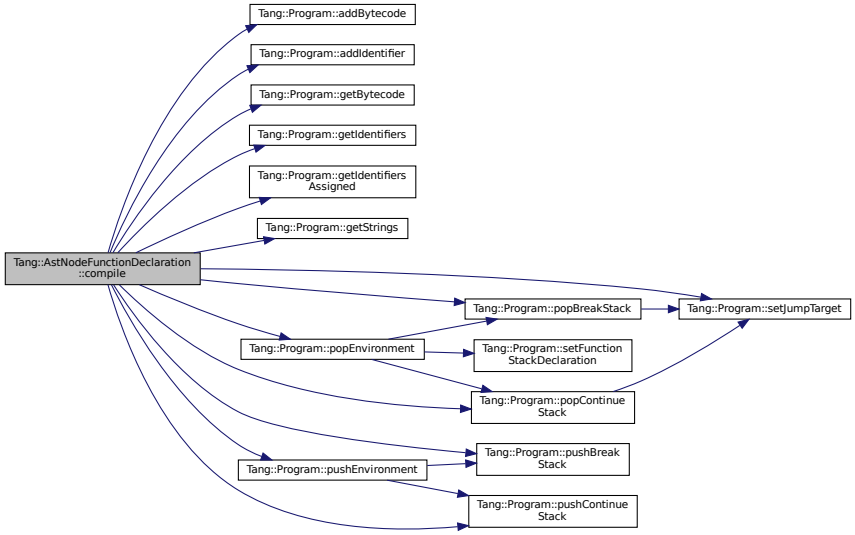
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.14.4.2 compilePreprocess()

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

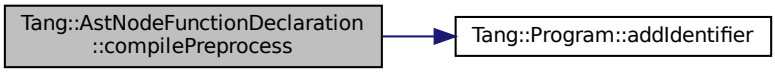
Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.14.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

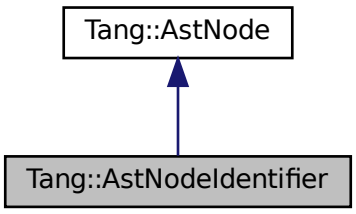
- include/astNodeFunctionDeclaration.hpp
- src/astNodeFunctionDeclaration.cpp

5.15 Tang::AstNodeIdentifier Class Reference

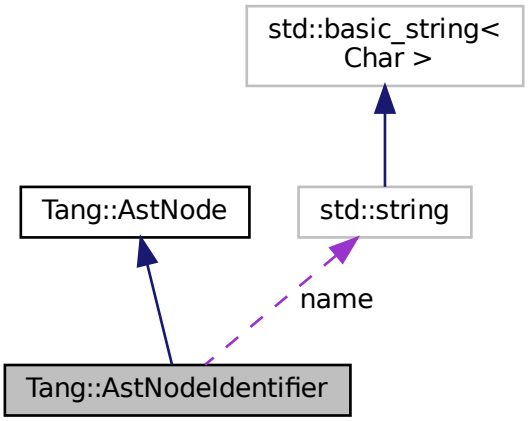
An AstNode that represents an identifier.

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

Inheritance diagram for Tang::AstNodeIdentifier:



Collaboration diagram for Tang::AstNodeIdentifier:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Public Attributes

- std::string name

  The name of the identifier.

5.15.1 Detailed Description

An AstNode that represents an identifier.

Identifier names are represented by a string.

5.15.2 Member Enumeration Documentation

5.15.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.15.3 Constructor & Destructor Documentation

5.15.3.1 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.

5.15.4 Member Function Documentation

5.15.4.1 compile()

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

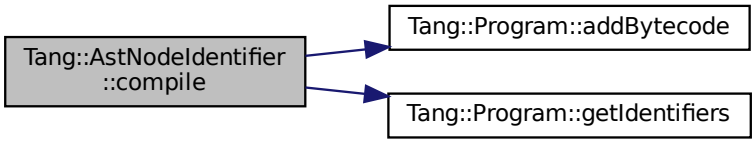
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.15.4.2 compilePreprocess()

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

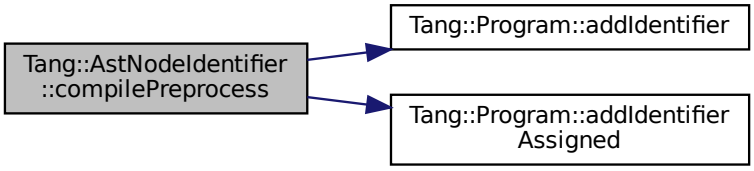
Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.15.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

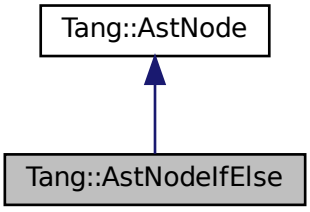
- include/astNodeIdentifier.hpp
- src/astNodeIdentifier.cpp

5.16 Tang::AstNodeIfElse Class Reference

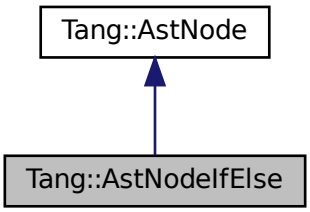
An AstNode that represents an if..else statement.

```
#include <astNodeIfElse.hpp>
```

Inheritance diagram for Tang::AstNodeIfElse:



Collaboration diagram for Tang::AstNodeIfElse:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

  The constructor.
- AstNodeIfElse (shared_ptr< AstNode > condition, shared_ptr< AstNode > thenBlock, Tang::location location)

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.16.1 Detailed Description

An AstNode that represents an if..else statement.

5.16.2 Member Enumeration Documentation

5.16.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.16.3 Constructor & Destructor Documentation

5.16.3.1 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.

5.16.3.2 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.

5.16.4 Member Function Documentation

5.16.4.1 compile()

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

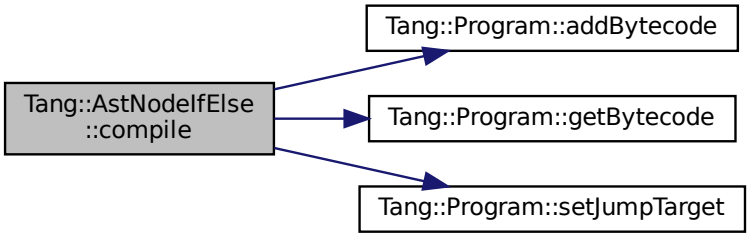
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.16.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.16.4.3 dump()

```
string AstNodeIfElse::dump (
```

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

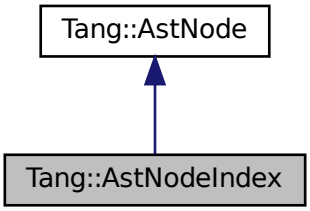
- include/astNodeIfElse.hpp
- src/astNodeIfElse.cpp

5.17 Tang::AstNodeIndex Class Reference

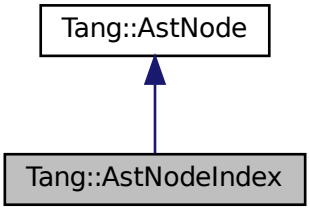
An AstNode that represents an index into a collection.

`#include <astNodeIndex.hpp>`

Inheritance diagram for Tang::AstNodeIndex:



Collaboration diagram for Tang::AstNodeIndex:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeIndex (std::shared_ptr< AstNode > collection, std::shared_ptr< AstNode > index, Tang::location location)

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.
- const std::shared_ptr< const AstNode > getCollection () const

  Return a shared pointer to the AstNode serving as the Collection.
- const std::shared_ptr< const AstNode > getIndex () const

  Return a shared pointer to the AstNode serving as the Index.

5.17.1 Detailed Description

An AstNode that represents an index into a collection.

5.17.2 Member Enumeration Documentation

5.17.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.17.3 Constructor & Destructor Documentation

5.17.3.1 AstNodeIndex()

```
AstNodeIndex::AstNodeIndex (
            std::shared_ptr< AstNode > collection,
            std::shared_ptr< AstNode > index,
            Tang::location location )
```

The constructor.

Parameters

collection	The collection into which we will index.
index	The index expression.
location	The location associated with the expression.

5.17.4 Member Function Documentation

5.17.4.1 compile()

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

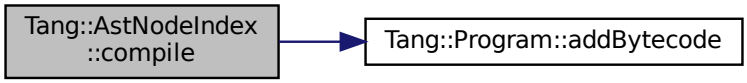
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.17.4.2 compilePreprocess()

```
void AstNodeIndex::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.17.4.3 dump()

```
string AstNodeIndex::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.

5.17.4.4 getCollection()

```
const std::shared_ptr< const AstNode > AstNodeIndex::getCollection ( ) const
```

Return a shared pointer to the AstNode serving as the Collection.

Returns

The collection into which we will index.

5.17.4.5 getIndex()

```
const std::shared_ptr< const AstNode > AstNodeIndex::getIndex ( ) const
```

Return a shared pointer to the AstNode serving as the Index.

Returns

The index expression.

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

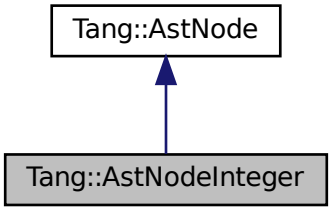
- include/astNodeIndex.hpp
- src/astNodeIndex.cpp

5.18 Tang::AstNodeInteger Class Reference

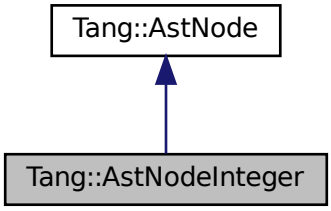
An AstNode that represents an integer literal.

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

Inheritance diagram for Tang::AstNodeInteger:



Collaboration diagram for Tang::AstNodeInteger:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const

  Run any preprocess analysis needed before compilation.

5.18.1 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.

5.18.2 Member Enumeration Documentation

5.18.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.18.3 Constructor & Destructor Documentation

5.18.3.1 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.

5.18.4 Member Function Documentation

5.18.4.1 compile()

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

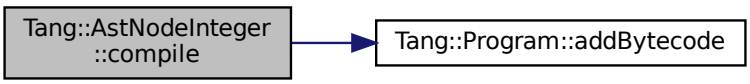
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.18.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

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

5.18.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

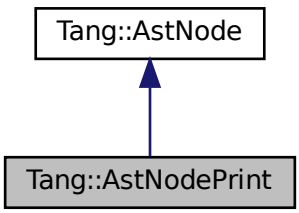
- include/astNodeInteger.hpp
- src/astNodeInteger.cpp

5.19 Tang::AstNodePrint Class Reference

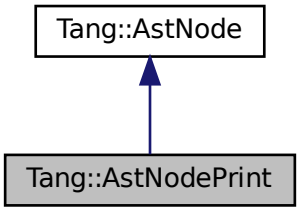
An AstNode that represents a print typeeration.

`#include <astNodePrint.hpp>`

Inheritance diagram for Tang::AstNodePrint:



Collaboration diagram for Tang::AstNodePrint:



Public Types

- enum Type { Default }

  The type of print() requested.
- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.19.1 Detailed Description

An AstNode that represents a print typeeration.

5.19.2 Member Enumeration Documentation

5.19.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.19.2.2 Type

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

The type of print() requested.

Enumerator

Default	Use the default print.

5.19.3 Constructor & Destructor Documentation

5.19.3.1 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.

5.19.4 Member Function Documentation

5.19.4.1 compile()

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

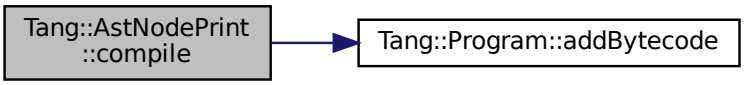
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.19.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.19.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

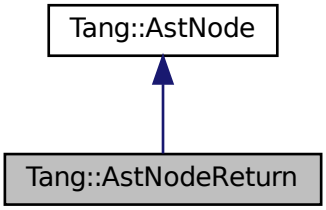
- include/astNodePrint.hpp
- src/astNodePrint.cpp

5.20 Tang::AstNodeReturn Class Reference

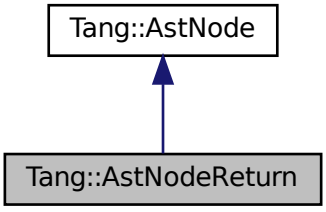
An AstNode that represents a `return` statement.

`#include <astNodeReturn.hpp>`

Inheritance diagram for Tang::AstNodeReturn:



Collaboration diagram for Tang::AstNodeReturn:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.20.1 Detailed Description

An AstNode that represents a `return` statement.

5.20.2 Member Enumeration Documentation

5.20.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.20.3 Constructor & Destructor Documentation

5.20.3.1 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.

5.20.4 Member Function Documentation

5.20.4.1 compile()

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

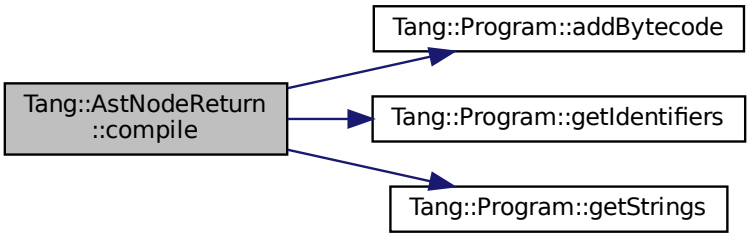
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.20.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.20.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

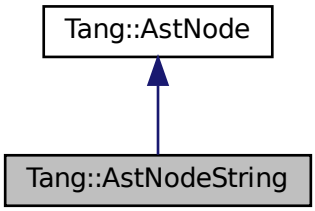
- include/astNodeReturn.hpp
- src/astNodeReturn.cpp

5.21 Tang::AstNodeString Class Reference

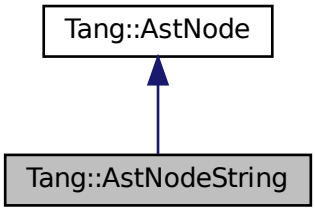
An AstNode that represents a string literal.

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

Inheritance diagram for Tang::AstNodeString:



Collaboration diagram for Tang::AstNodeString:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

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

  Compile the string and push it onto the stack.

5.21.1 Detailed Description

An AstNode that represents a string literal.

5.21.2 Member Enumeration Documentation

5.21.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.21.3 Constructor & Destructor Documentation

5.21.3.1 AstNodeString()

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

The constructor.

Parameters

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

5.21.4 Member Function Documentation

5.21.4.1 compile()

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

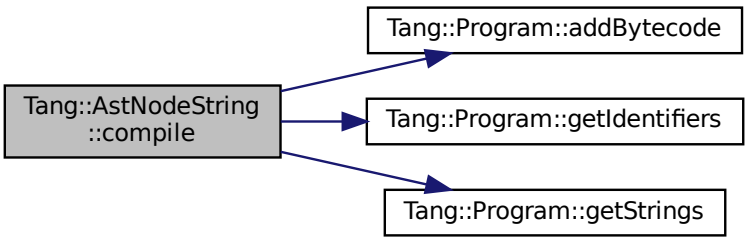
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.21.4.2 compileLiteral()

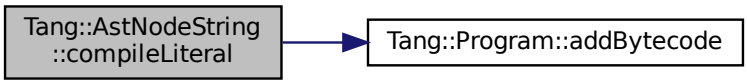
```
void AstNodeString::compileLiteral (
            Tang::Program & program ) const
```

Compile the string and push it onto the stack.

Parameters

program	The Program which will hold the generated Bytecode.

Here is the call graph for this function:



5.21.4.3 compilePreprocess()

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

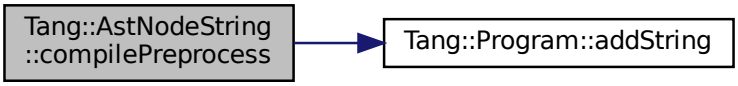
Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.21.4.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

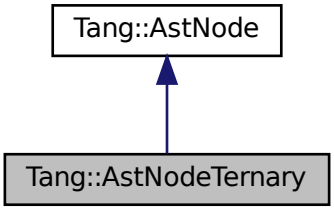
- include/astNodeString.hpp
- src/astNodeString.cpp

5.22 Tang::AstNodeTernary Class Reference

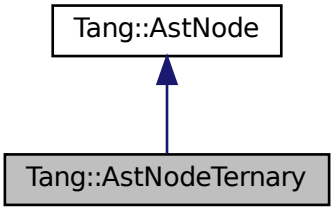
An AstNode that represents a ternary expression.

`#include <astNodeTernary.hpp>`

Inheritance diagram for Tang::AstNodeTernary:



Collaboration diagram for Tang::AstNodeTernary:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.22.1 Detailed Description

An AstNode that represents a ternary expression.

5.22.2 Member Enumeration Documentation

5.22.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.22.3 Constructor & Destructor Documentation

5.22.3.1 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.

5.22.4 Member Function Documentation

5.22.4.1 compile()

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

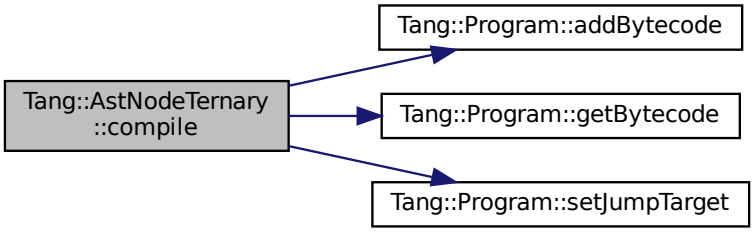
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.22.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.22.4.3 dump()

```
string AstNodeTernary::dump (
```

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

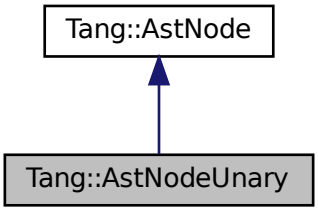
- include/astNodeTernary.hpp
- src/astNodeTernary.cpp

5.23 Tang::AstNodeUnary Class Reference

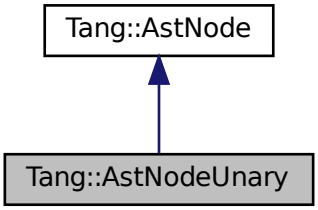
An AstNode that represents a unary negation.

`#include <astNodeUnary.hpp>`

Inheritance diagram for Tang::AstNodeUnary:



Collaboration diagram for Tang::AstNodeUnary:



Public Types

- enum Operator { Negative , Not }

  The type of operation.
- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.23.1 Detailed Description

An AstNode that represents a unary negation.

5.23.2 Member Enumeration Documentation

5.23.2.1 Operator

`enum Tang::AstNodeUnary::Operator`

The type of operation.

Enumerator

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

5.23.2.2 PreprocessState

`enum Tang::AstNode::PreprocessState : int [inherited]`

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.23.3 Constructor & Destructor Documentation

5.23.3.1 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.

5.23.4 Member Function Documentation

5.23.4.1 compile()

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

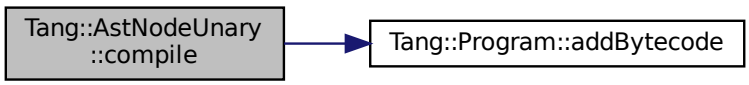
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.23.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.23.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

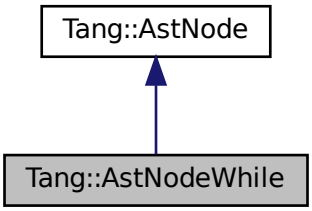
- include/astNodeUnary.hpp
- src/astNodeUnary.cpp

5.24 Tang::AstNodeWhile Class Reference

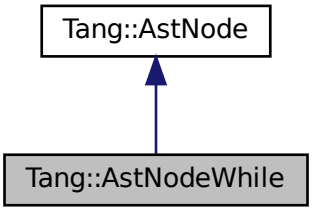
An AstNode that represents a while statement.

`#include <astNodeWhile.hpp>`

Inheritance diagram for Tang::AstNodeWhile:



Collaboration diagram for Tang::AstNodeWhile:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

5.24.1 Detailed Description

An AstNode that represents a while statement.

5.24.2 Member Enumeration Documentation

5.24.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.24.3 Constructor & Destructor Documentation

5.24.3.1 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.

5.24.4 Member Function Documentation

5.24.4.1 compile()

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

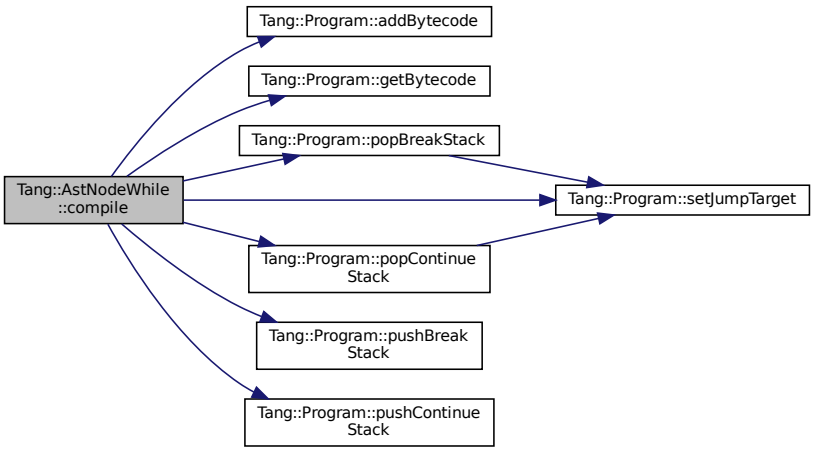
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.24.4.2 compilePreprocess()

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

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.24.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

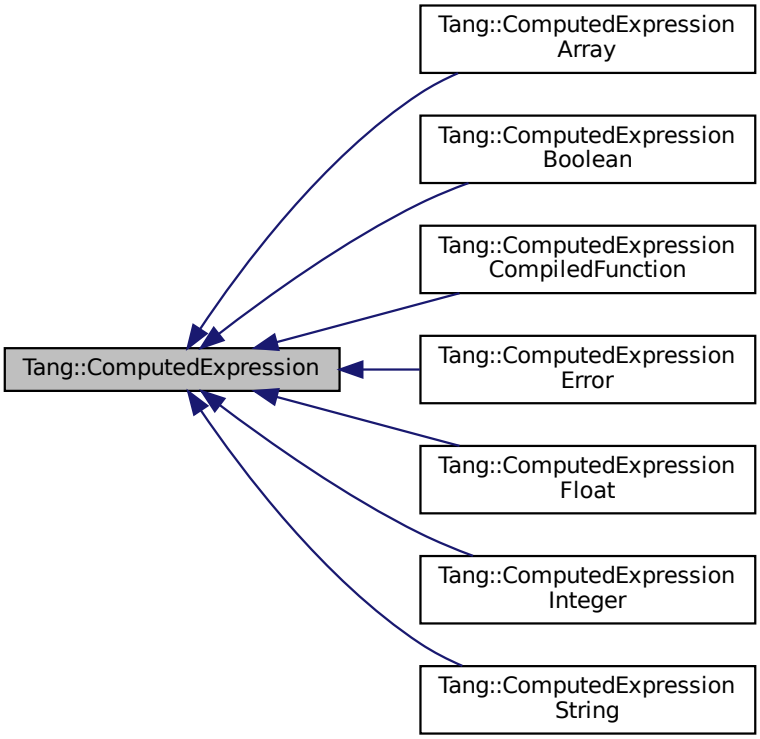
- include/astNodeWhile.hpp
- src/astNodeWhile.cpp

5.25 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 bool isCopyNeeded () const

  Determine whether or not a copy is needed.
- 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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)

  Perform an index assignment to the supplied 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 equality test.
- virtual GarbageCollected __index (const GarbageCollected &index) const

  Perform an index operation.
- 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.

5.25.1 Detailed Description

Represents the result of a computation that has been executed.

By default, it will represent a NULL value.

5.25.2 Member Function Documentation

5.25.2.1 __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.

5.25.2.2 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.25.2.3 __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.

5.25.2.4 __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.

5.25.2.5 __equal()

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

Perform an equality 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.

5.25.2.6 __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.

5.25.2.7 __index()

```
GarbageCollected ComputedExpression::__index (
            const GarbageCollected & index ) const  [virtual]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.25.2.8 __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.

5.25.2.9 __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.

5.25.2.10 __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.

5.25.2.11 __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.

5.25.2.12 __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.

5.25.2.13 __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.

5.25.2.14 __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.

5.25.2.15 __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.

5.25.2.16 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, Tang::ComputedExpressionBoolean, and Tang::ComputedExpressionArray.

5.25.2.17 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.

5.25.2.18 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.

5.25.2.19 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.

5.25.2.20 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.

5.25.2.21 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.

5.25.2.22 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.

5.25.2.23 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.25.2.24 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, Tang::ComputedExpressionBoolean, and Tang::ComputedExpressionArray.

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

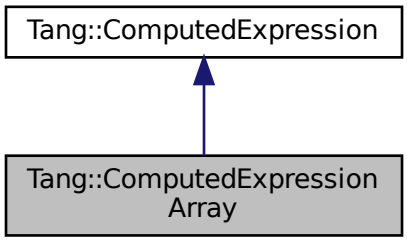
- include/computedExpression.hpp
- src/computedExpression.cpp

5.26 Tang::ComputedExpressionArray Class Reference

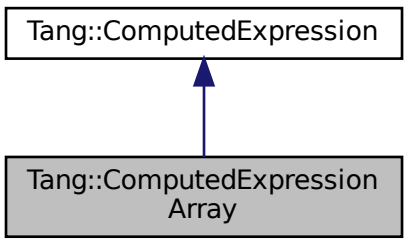
Represents an Array that is the result of a computation.

`#include <computedExpressionArray.hpp>`

Inheritance diagram for Tang::ComputedExpressionArray:



Collaboration diagram for Tang::ComputedExpressionArray:



Public Member Functions

- ComputedExpressionArray (std::vector< Tang::GarbageCollected > contents)

  Construct an Array result.
- virtual std::string dump () const override

  Output the contents of the ComputedExpression as a string.
- virtual bool isCopyNeeded () const override

  Determine whether or not a copy is needed.
- GarbageCollected makeCopy () const override

  Make a copy of the ComputedExpression (recursively, if appropriate).
- virtual GarbageCollected __index (const GarbageCollected &index) const override

  Perform an index operation.
- virtual GarbageCollected __assign_index (const GarbageCollected &index, const GarbageCollected &value) override

  Perform an index assignment to the supplied value.
- 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 equality 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.

5.26.1 Detailed Description

Represents an Array that is the result of a computation.

5.26.2 Constructor & Destructor Documentation

5.26.2.1 ComputedExpressionArray()

```
ComputedExpressionArray::ComputedExpressionArray (
            std::vector< Tang::GarbageCollected > contents )
```

Construct an Array result.

Parameters

val	The integer value.

5.26.3 Member Function Documentation

5.26.3.1 __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.

5.26.3.2 __assign_index()

```
GarbageCollected ComputedExpressionArray::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [override], [virtual]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.26.3.3 __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.

5.26.3.4 __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.

5.26.3.5 __equal()

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

Perform an equality 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.

5.26.3.6 __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.

5.26.3.7 __index()

```
GarbageCollected ComputedExpressionArray::__index (
            const GarbageCollected & index ) const  [override], [virtual]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.26.3.8 __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.

5.26.3.9 __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.

5.26.3.10 __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.

5.26.3.11 __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.

5.26.3.12 __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.

5.26.3.13 __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.

5.26.3.14 __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.

5.26.3.15 __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.

5.26.3.16 dump()

```
string ComputedExpressionArray::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.

5.26.3.17 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.

5.26.3.18 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.

5.26.3.19 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.

5.26.3.20 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.

5.26.3.21 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.

5.26.3.22 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.

5.26.3.23 isCopyNeeded()

```
bool ComputedExpressionArray::isCopyNeeded ( ) const  [override], [virtual]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented from Tang::ComputedExpression.

5.26.3.24 makeCopy()

```
GarbageCollected ComputedExpressionArray::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:

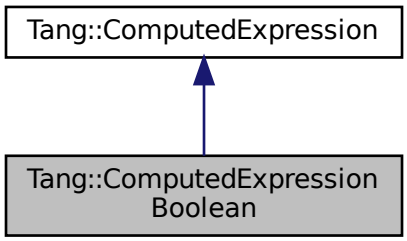
- include/computedExpressionArray.hpp
- src/computedExpressionArray.cpp

5.27 Tang::ComputedExpressionBoolean Class Reference

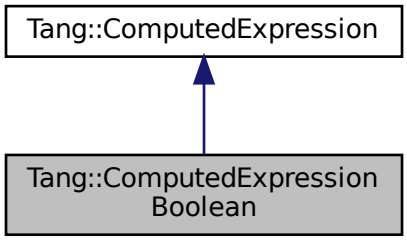
Represents an Boolean that is the result of a computation.

`#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 equality test.
- virtual GarbageCollected __integer () const override

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

  Perform a type cast to float.
- virtual GarbageCollected __boolean () const override

  Perform a type cast to boolean.
- virtual bool isCopyNeeded () const

  Determine whether or not a copy is needed.
- 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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)

  Perform an index assignment to the supplied 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 __index (const GarbageCollected &index) const

  Perform an index operation.
- virtual GarbageCollected __string () const

  Perform a type cast to string.

5.27.1 Detailed Description

Represents an Boolean that is the result of a computation.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 ComputedExpressionBoolean()

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

Construct an Boolean result.

Parameters

val	The boolean value.

5.27.3 Member Function Documentation

5.27.3.1 __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.

5.27.3.2 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.27.3.3 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.27.3.4 __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.

5.27.3.5 __equal()

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

Perform an equality test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.27.3.6 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.27.3.7 __index()

```
GarbageCollected ComputedExpression::__index (
            const GarbageCollected & index ) const  [virtual], [inherited]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.27.3.8 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.27.3.9 __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.

5.27.3.10 __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.

5.27.3.11 __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.

5.27.3.12 __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.

5.27.3.13 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.27.3.14 __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.

5.27.3.15 __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.

5.27.3.16 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.

5.27.3.17 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.

5.27.3.18 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.

5.27.3.19 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.

5.27.3.20 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.

5.27.3.21 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.

5.27.3.22 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.

5.27.3.23 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.27.3.24 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:

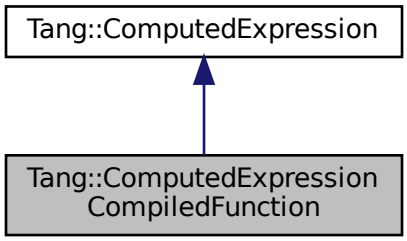
- include/computedExpressionBoolean.hpp
- src/computedExpressionBoolean.cpp

5.28 Tang::ComputedExpressionCompiledFunction Class Reference

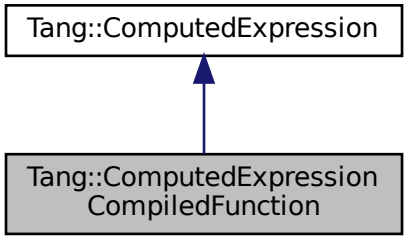
Represents a Compiled Function declared in the script.

`#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 equality test.
- uint32_t getArgc () const

  Get the `argc` value.

- Tang::integer_t getPc () const

  Get the bytecode target.
- virtual bool isCopyNeeded () const

  Determine whether or not a copy is needed.
- 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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)

  Perform an index assignment to the supplied 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 __index (const GarbageCollected &index) const

  Perform an index operation.
- 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.

5.28.1 Detailed Description

Represents a Compiled Function declared in the script.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 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.

5.28.3 Member Function Documentation

5.28.3.1 __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.

5.28.3.2 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.28.3.3 __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.

5.28.3.4 __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.

5.28.3.5 __equal()

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

Perform an equality test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.28.3.6 __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.

5.28.3.7 __index()

```
GarbageCollected ComputedExpression::__index (
            const GarbageCollected & index ) const  [virtual], [inherited]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.28.3.8 __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.

5.28.3.9 __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.

5.28.3.10 __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.

5.28.3.11 __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.

5.28.3.12 __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.

5.28.3.13 __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.

5.28.3.14 __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.

5.28.3.15 __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.

5.28.3.16 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.

5.28.3.17 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.

5.28.3.18 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.

5.28.3.19 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.

5.28.3.20 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.

5.28.3.21 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.

5.28.3.22 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.

5.28.3.23 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.28.3.24 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:

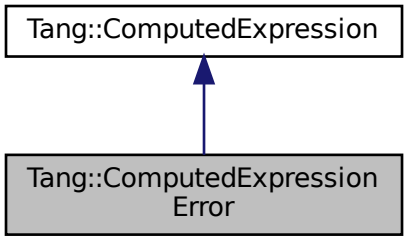
- include/computedExpressionCompiledFunction.hpp
- src/computedExpressionCompiledFunction.cpp

5.29 Tang::ComputedExpressionError Class Reference

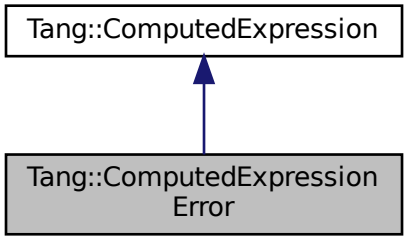
Represents a Runtime Error.

`#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 equality test.
- virtual GarbageCollected __integer () const override

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

  Perform a type cast to float.
- virtual GarbageCollected __boolean () const override

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

  Perform a type cast to string.
- virtual bool isCopyNeeded () const

  Determine whether or not a copy is needed.
- 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.
- virtual GarbageCollected __assign_index (const GarbageCollected &index, const GarbageCollected &value)

  Perform an index assignment to the supplied value.
- virtual GarbageCollected __index (const GarbageCollected &index) const

  Perform an index operation.

5.29.1 Detailed Description

Represents a Runtime Error.

5.29.2 Constructor & Destructor Documentation

5.29.2.1 ComputedExpressionError()

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

Construct a Runtime Error.

Parameters

error	The Tang::Error object.

5.29.3 Member Function Documentation

5.29.3.1 __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.

5.29.3.2 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.29.3.3 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.29.3.4 __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.

5.29.3.5 __equal()

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

Perform an equality test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.29.3.6 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.29.3.7 __index()

```
GarbageCollected ComputedExpression::__index (
            const GarbageCollected & index ) const  [virtual], [inherited]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.29.3.8 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.29.3.9 __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.

5.29.3.10 __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.

5.29.3.11 __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.

5.29.3.12 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.29.3.13 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.29.3.14 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.29.3.15 __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.

5.29.3.16 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.

5.29.3.17 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.

5.29.3.18 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.

5.29.3.19 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.

5.29.3.20 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.

5.29.3.21 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.

5.29.3.22 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.

5.29.3.23 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.29.3.24 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:

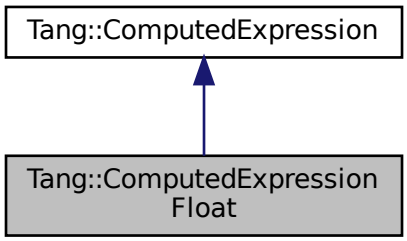
- include/computedExpressionError.hpp
- src/computedExpressionError.cpp

5.30 Tang::ComputedExpressionFloat Class Reference

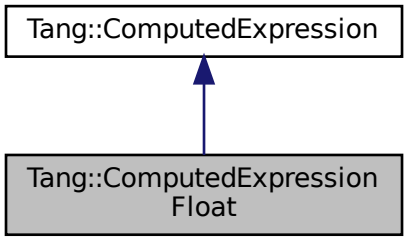
Represents a Float that is the result of a computation.

`#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 equality test.
- virtual GarbageCollected __integer () const override

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

  Perform a type cast to float.
- virtual GarbageCollected __boolean () const override

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

  Perform a type cast to string.
- virtual bool isCopyNeeded () const

  Determine whether or not a copy is needed.
- 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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)

  Perform an index assignment to the supplied value.
- virtual GarbageCollected __modulo (const GarbageCollected &rhs) const

  Compute the result of moduloing this value and the supplied value.
- virtual GarbageCollected __index (const GarbageCollected &index) const

  Perform an index operation.

Friends

- class **ComputedExpressionInteger**

5.30.1 Detailed Description

Represents a Float that is the result of a computation.

5.30.2 Constructor & Destructor Documentation

5.30.2.1 ComputedExpressionFloat()

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

Construct a Float result.

Parameters

val	The float value.

5.30.3 Member Function Documentation

5.30.3.1 __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.

5.30.3.2 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.30.3.3 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.30.3.4 __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.

5.30.3.5 __equal()

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

Perform an equality test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.30.3.6 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.30.3.7 __index()

```
GarbageCollected ComputedExpression::__index (
            const GarbageCollected & index ) const  [virtual], [inherited]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.30.3.8 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.30.3.9 __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.

5.30.3.10 __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.

5.30.3.11 __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.

5.30.3.12 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.30.3.13 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.30.3.14 __string()

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

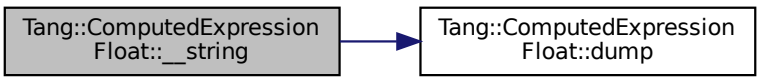
Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.30.3.15 __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.

5.30.3.16 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.

5.30.3.17 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.

5.30.3.18 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.

5.30.3.19 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.

5.30.3.20 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.

5.30.3.21 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.

5.30.3.22 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.

5.30.3.23 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.30.3.24 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:

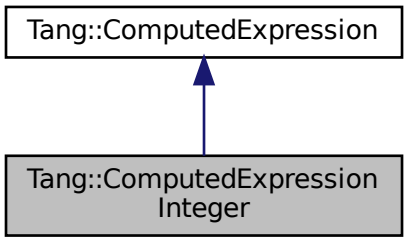
- include/computedExpressionFloat.hpp
- src/computedExpressionFloat.cpp

5.31 Tang::ComputedExpressionInteger Class Reference

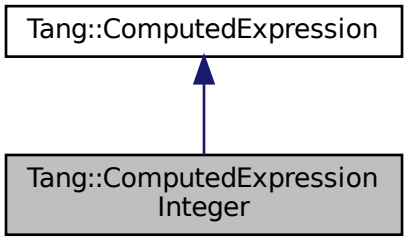
Represents an Integer that is the result of a computation.

`#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 equality test.
- virtual GarbageCollected __integer () const override

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

  Perform a type cast to float.
- virtual GarbageCollected __boolean () const override

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

  Perform a type cast to string.
- virtual bool isCopyNeeded () const

  Determine whether or not a copy is needed.
- 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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)

  Perform an index assignment to the supplied value.
- virtual GarbageCollected __index (const GarbageCollected &index) const

  Perform an index operation.

Friends

- class **ComputedExpressionFloat**
- class **ComputedExpressionArray**

5.31.1 Detailed Description

Represents an Integer that is the result of a computation.

5.31.2 Constructor & Destructor Documentation

5.31.2.1 ComputedExpressionInteger()

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

Construct an Integer result.

Parameters

val	The integer value.

5.31.3 Member Function Documentation

5.31.3.1 __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.

5.31.3.2 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.31.3.3 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.31.3.4 __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.

5.31.3.5 __equal()

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

Perform an equality test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.31.3.6 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.31.3.7 __index()

```
GarbageCollected ComputedExpression::__index (
            const GarbageCollected & index ) const  [virtual], [inherited]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.31.3.8 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.31.3.9 __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.

5.31.3.10 __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.

5.31.3.11 __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.

5.31.3.12 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.31.3.13 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.31.3.14 __string()

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

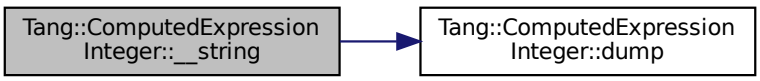
Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.31.3.15 __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.

5.31.3.16 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.

5.31.3.17 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.

5.31.3.18 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.

5.31.3.19 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.

5.31.3.20 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.

5.31.3.21 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.

5.31.3.22 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.

5.31.3.23 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.31.3.24 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:

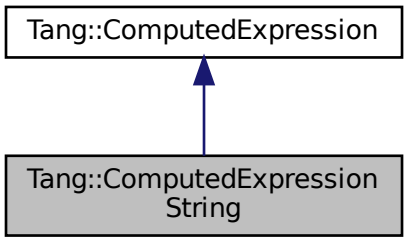
- include/computedExpressionInteger.hpp
- src/computedExpressionInteger.cpp

5.32 Tang::ComputedExpressionString Class Reference

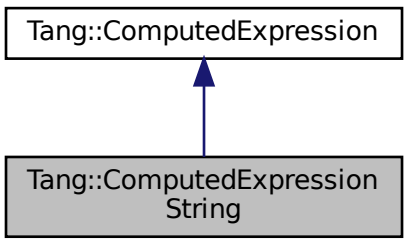
Represents a String that is the result of a computation.

`#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 equality 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 isCopyNeeded () const

  Determine whether or not a copy is needed.
- 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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)

  Perform an index assignment to 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 __index (const GarbageCollected &index) const

  Perform an index operation.
- virtual GarbageCollected __integer () const

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

  Perform a type cast to float.

5.32.1 Detailed Description

Represents a String that is the result of a computation.

5.32.2 Constructor & Destructor Documentation

5.32.2.1 ComputedExpressionString()

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

Construct a String result.

Parameters

val	The string value.

5.32.3 Member Function Documentation

5.32.3.1 __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.

5.32.3.2 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.32.3.3 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.32.3.4 __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.

5.32.3.5 __equal()

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

Perform an equality test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.32.3.6 __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.

5.32.3.7 __index()

```
GarbageCollected ComputedExpression::__index (
            const GarbageCollected & index ) const  [virtual], [inherited]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.32.3.8 __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.

5.32.3.9 __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.

5.32.3.10 __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.

5.32.3.11 __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.

5.32.3.12 __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.

5.32.3.13 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.32.3.14 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.32.3.15 __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.

5.32.3.16 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.

5.32.3.17 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.

5.32.3.18 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.

5.32.3.19 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.

5.32.3.20 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.

5.32.3.21 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.

5.32.3.22 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.

5.32.3.23 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.32.3.24 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:

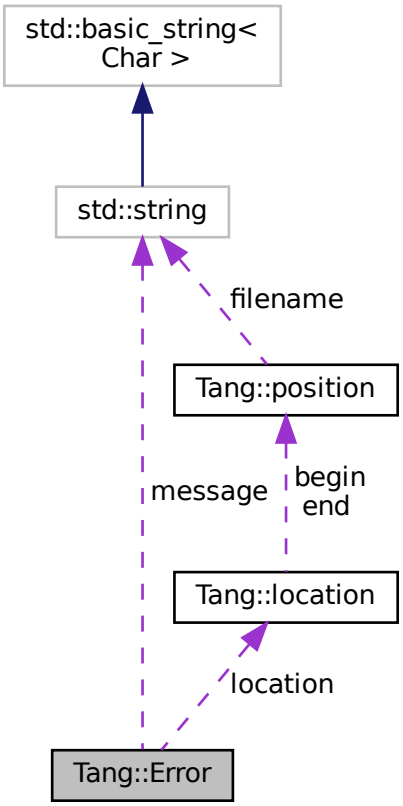
- include/computedExpressionString.hpp
- src/computedExpressionString.cpp

5.33 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.

5.33.1 Detailed Description

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

5.33.2 Constructor & Destructor Documentation

5.33.2.1 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.

5.33.2.2 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.

5.33.3 Friends And Related Function Documentation

5.33.3.1 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:

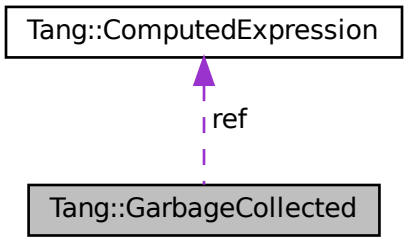
- include/error.hpp
- src/error.cpp

5.34 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.
- bool isCopyNeeded () const

  Determine whether or not a copy is needed as determined by the referenced ComputedExpression.
- GarbageCollected makeCopy () const

  Create a separate copy of the original GarbageCollected value.
- 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.

5.34.1 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.

5.34.2 Constructor & Destructor Documentation

5.34.2.1 GarbageCollected() [1/3]

```
GarbageCollected::GarbageCollected (
            const GarbageCollected & other )
```

Copy Constructor.

Parameters

The	other GarbageCollected object to copy.

5.34.2.2 GarbageCollected() [2/3]

```
GarbageCollected::GarbageCollected (
            GarbageCollected && other )
```

Move Constructor.

Parameters

The	other GarbageCollected object to move.

5.34.2.3 ~GarbageCollected()

```
GarbageCollected::~GarbageCollected ( )
```

Destructor.

Clean up the tracked object, if appropriate.

5.34.2.4 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.

5.34.3 Member Function Documentation

5.34.3.1 isCopyNeeded()

```
bool GarbageCollected::isCopyNeeded ( ) const
```

Determine whether or not a copy is needed as determined by the referenced ComputedExpression.

Returns

Whether or not a copy is needed.

5.34.3.2 make()

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

Creates a garbage-collected object of the specified type.

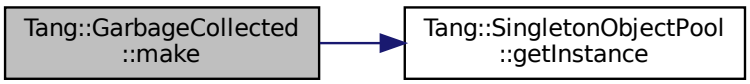
Parameters

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

Returns

A GarbageCollected object.

Here is the call graph for this function:



5.34.3.3 makeCopy()

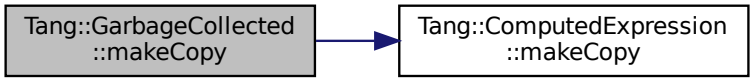
```
GarbageCollected GarbageCollected::makeCopy ( ) const
```

Create a separate copy of the original GarbageCollected value.

Returns

A GarbageCollected copy of the original value.

Here is the call graph for this function:



5.34.3.4 operator"!()

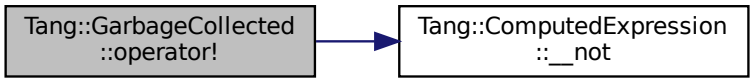
```
GarbageCollected GarbageCollected::operator!  ( ) const
```

Perform a logical not on the GarbageCollected value.

Returns

The result of the operation.

Here is the call graph for this function:



5.34.3.5 operator"!=()

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

Perform a != between two GarbageCollected values.

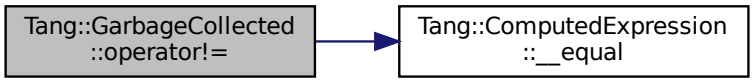
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.34.3.6 operator%()

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

Perform a modulo between two GarbageCollected values.

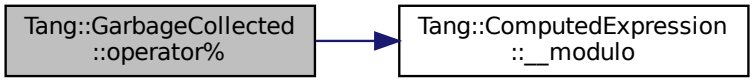
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.34.3.7 operator*() [1/2]

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

Access the tracked object.

Returns

A reference to the tracked object.

5.34.3.8 operator*() [2/2]

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

Perform a multiplication between two GarbageCollected values.

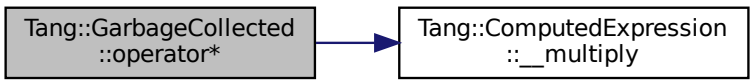
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.34.3.9 operator+()

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

Perform an addition between two GarbageCollected values.

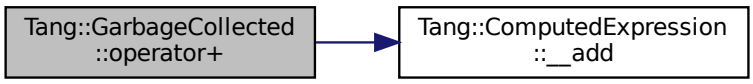
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.34.3.10 operator-() [1/2]

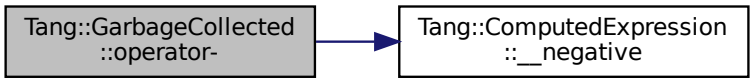
```
GarbageCollected GarbageCollected::operator- ( ) const
```

Perform a negation on the GarbageCollected value.

Returns

The result of the operation.

Here is the call graph for this function:



5.34.3.11 operator-() [2/2]

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

Perform a subtraction between two GarbageCollected values.

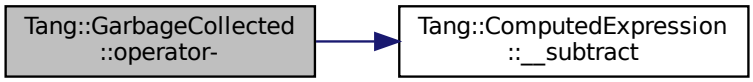
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.34.3.12 operator->()

```
ComputedExpression * GarbageCollected::operator-> ( ) const
```

Access the tracked object as a pointer.

Returns

A pointer to the tracked object.

5.34.3.13 operator/()

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

Perform a division between two GarbageCollected values.

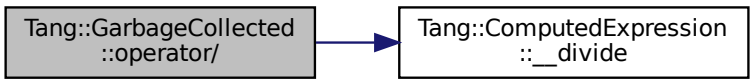
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.34.3.14 operator<()

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

Perform a < between two GarbageCollected values.

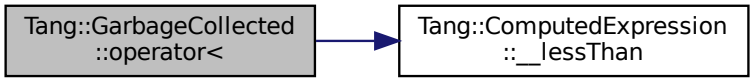
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.34.3.15 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.

5.34.3.16 operator=() [1/2]

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

Copy Assignment.

Parameters

The	other GarbageCollected object.

5.34.3.17 operator=() [2/2]

```
GarbageCollected & GarbageCollected::operator= (
            GarbageCollected && other )
```

Move Assignment.

Parameters

The	other GarbageCollected object.

5.34.3.18 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.

5.34.3.19 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.

5.34.3.20 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.

5.34.3.21 operator==() [4/8]

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

Perform a == between two GarbageCollected values.

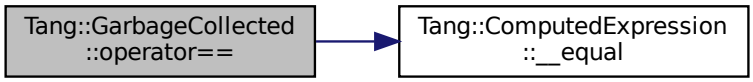
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.34.3.22 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.

5.34.3.23 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.

5.34.3.24 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.

5.34.3.25 operator==() [8/8]

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

Compare the GarbageCollected tracked object with a supplied value.

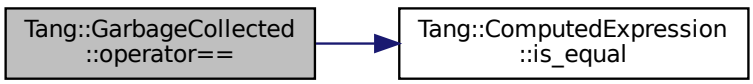
Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

Here is the call graph for this function:



5.34.3.26 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.

5.34.3.27 operator>=()

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

Perform a >= between two GarbageCollected values.

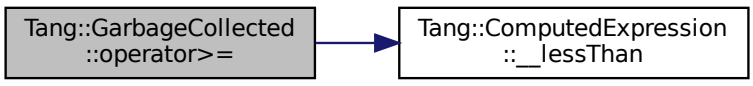
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.34.4 Friends And Related Function Documentation

5.34.4.1 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:

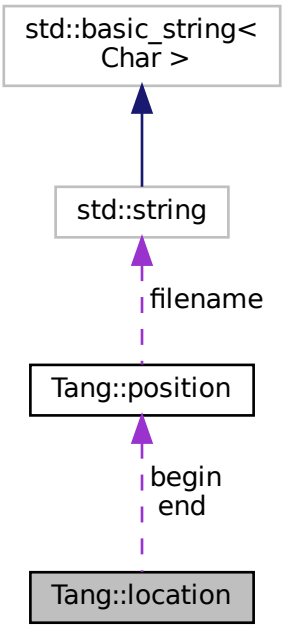
- include/garbageCollected.hpp
- src/garbageCollected.cpp

5.35 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.

5.35.1 Detailed Description

Two points in a source file.

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

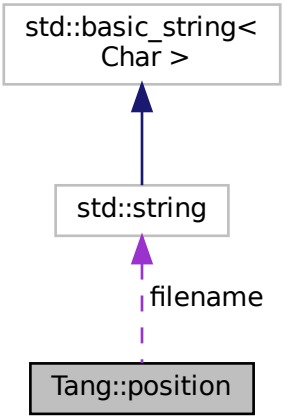
- build/generated/location.hh

5.36 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.

5.36.1 Detailed Description

A point in a source file.

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

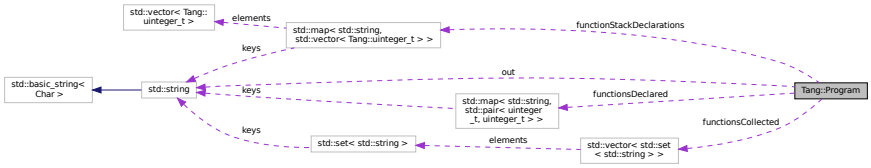
- build/generated/location.hh

5.37 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 addIdentifierAssigned (const std::string &name)

  Indicate that an identifier will be altered within the associated scope.
- const std::set< std::string > & getIdentifiersAssigned () const

  Get the set of identifiers that will be assigned in the current scope.
- 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.
- void pushBreakStack ()

  Increase the `break` environment stack, so that we can handle nested break-supporting structures.
- void addBreak (size_t location)

  Add the Bytecode location of a `break` statement, to be set when the final target is known at a later time.
- void popBreakStack (size_t target)

  For all `continue` bytecode locations collected by Tang::addContinue, set the target pc to `target`.
- void pushContinueStack ()

  Increase the `continue` environment stack, so that we can handle nested continue-supporting structures.
- void addContinue (size_t location)

  Add the Bytecode location of a `continue` statement, to be set when the final target is known at a later time.
- void popContinueStack (size_t target)

  For all `continue` bytecode locations collected by Tang::addContinue, set the target pc to `target`.

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.

5.37.1 Detailed Description

Represents a compiled script or template that may be executed.

5.37.2 Member Enumeration Documentation

5.37.2.1 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.

5.37.3 Constructor & Destructor Documentation

5.37.3.1 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`.

5.37.4 Member Function Documentation

5.37.4.1 addBreak()

```
void Program::addBreak (
            size_t location )
```

Add the Bytecode location of a `break` statement, to be set when the final target is known at a later time.

Parameters

location	The offset location of the `break` bytecode.

5.37.4.2 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.

5.37.4.3 addContinue()

```
void Program::addContinue (
            size_t location )
```

Add the Bytecode location of a `continue` statement, to be set when the final target is known at a later time.

Parameters

location	The offset location of the `continue` bytecode.

5.37.4.4 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.

5.37.4.5 addIdentifierAssigned()

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

Indicate that an identifier will be altered within the associated scope.

Parameters

name	The identifier name.

5.37.4.6 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.

5.37.4.7 dumpBytecode()

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

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

Returns

A string containing the Opcode representation.

5.37.4.8 execute()

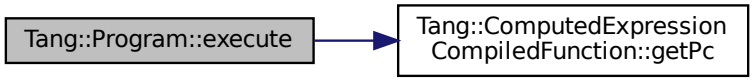
```
Program & Program::execute ( )
```

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

Returns

The current Program object.

Here is the call graph for this function:



5.37.4.9 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.

5.37.4.10 getBytecode()

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

Get the Bytecode vector.

Returns

The Bytecode vector.

5.37.4.11 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.

5.37.4.12 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.

5.37.4.13 getIdentifiersAssigned()

```
const set< string > & Program::getIdentifiersAssigned ( ) const
```

Get the set of identifiers that will be assigned in the current scope.

Returns

A set of identifier names that have been identified as the target of an assignment operator within the current scope.

5.37.4.14 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.

5.37.4.15 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.

5.37.4.16 popBreakStack()

```
void Program::popBreakStack (
            size_t target )
```

For all `continue` bytecode locations collected by Tang::addContinue, set the target pc to `target`.

Parameters

target	The target bytecode offset that the `continue` should jump to.

Here is the call graph for this function:

5.37.4.17 popContinueStack()

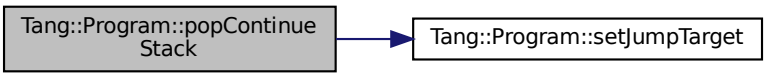
```
void Program::popContinueStack (
            size_t target )
```

For all `continue` bytecode locations collected by Tang::addContinue, set the target pc to `target`.

Parameters

target	The target bytecode offset that the `continue` should jump to.

Here is the call graph for this function:



5.37.4.18 pushEnvironment()

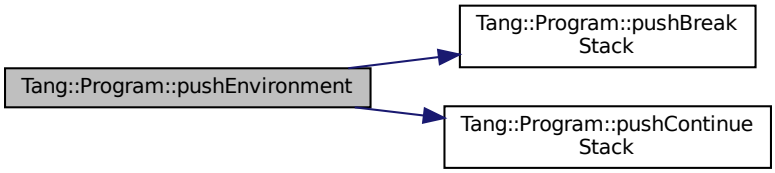
```
void Program::pushEnvironment (
            const std::shared_ptr< AstNode > & ast )
```

Create a new compile/execute environment stack entry.

Parameters

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

Here is the call graph for this function:



5.37.4.19 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.

5.37.4.20 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.

5.37.5 Member Data Documentation

5.37.5.1 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

5.38 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.

5.38.1 Detailed Description

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

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

5.38.2 Member Function Documentation

5.38.2.1 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.

5.38.2.2 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.

5.38.2.3 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

5.39 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.

5.39.1 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).

5.39.2 Constructor & Destructor Documentation

5.39.2.1 TangBase()

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

The constructor.

Isn't it glorious.

5.39.3 Member Function Documentation

5.39.3.1 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:

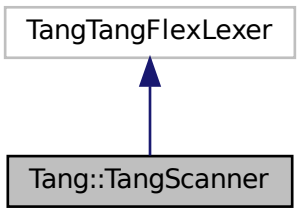
- include/tangBase.hpp
- src/tangBase.cpp

5.40 Tang::TangScanner Class Reference

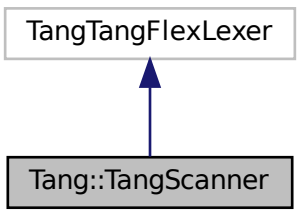
The Flex lexer class for the main Tang language.

`#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.

5.40.1 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.

5.40.2 Constructor & Destructor Documentation

5.40.2.1 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)

5.40.3 Member Function Documentation

5.40.3.1 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

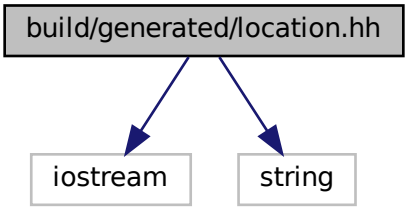
File Documentation

6.1 build/generated/location.hh File Reference

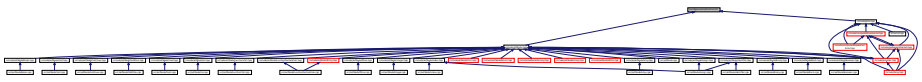
Define the Tang ::location class.

```
#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

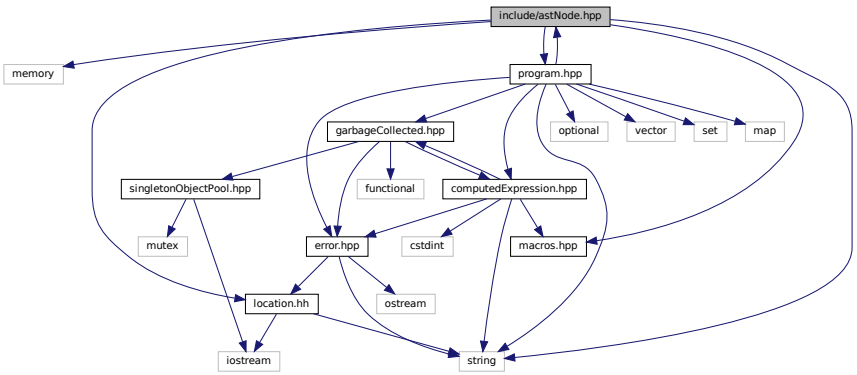
ostr	the destination output stream
pos	a reference to the position to redirect

6.2 include/astNode.hpp File Reference

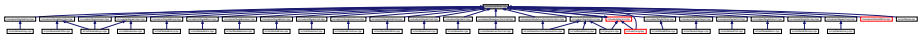
Declare the Tang::AstNode base class.

```
#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

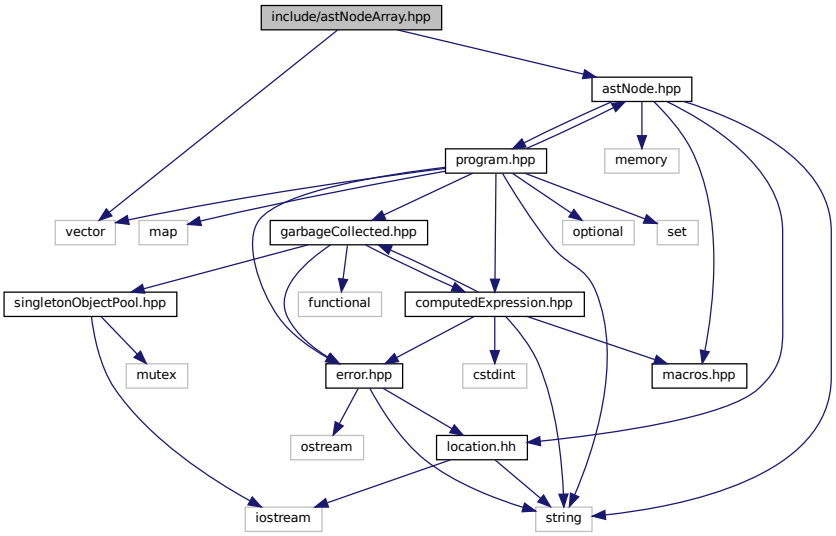
Declare the Tang::AstNode base class.

6.3 include/astNodeArray.hpp File Reference

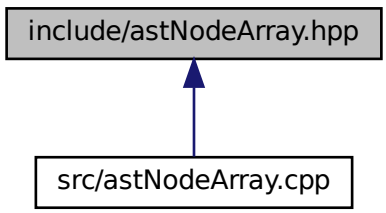
Declare the Tang::AstNodeArray class.

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

Include dependency graph for astNodeArray.hpp:



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



Classes

- class Tang::AstNodeArray

  An AstNode that represents an array literal.

6.3.1 Detailed Description

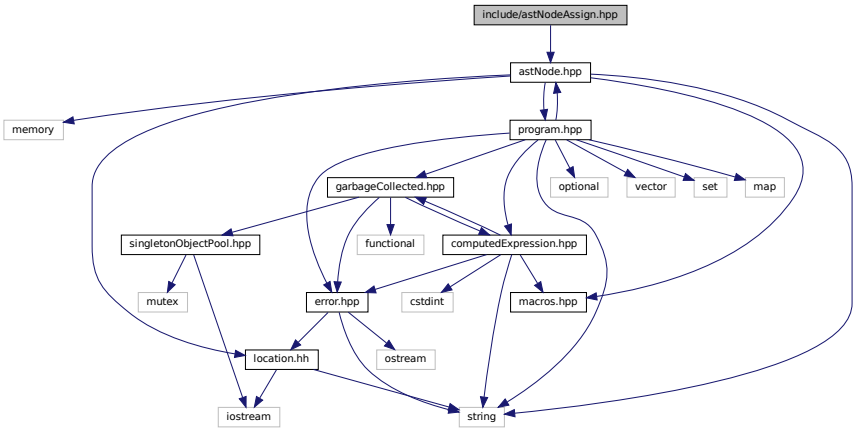
Declare the Tang::AstNodeArray class.

6.4 include/astNodeAssign.hpp File Reference

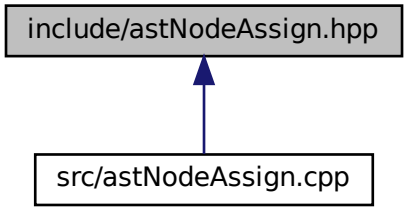
Declare the Tang::AstNodeAssign class.

```
#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.

6.4.1 Detailed Description

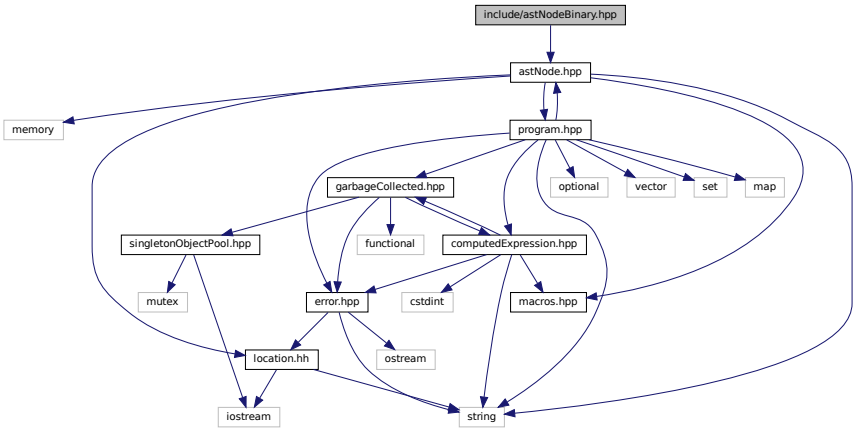
Declare the Tang::AstNodeAssign class.

6.5 include/astNodeBinary.hpp File Reference

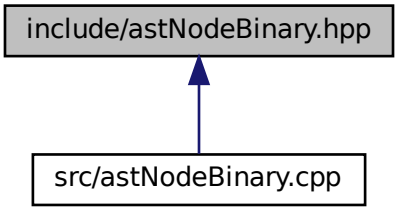
Declare the Tang::AstNodeBinary class.

```
#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.

6.5.1 Detailed Description

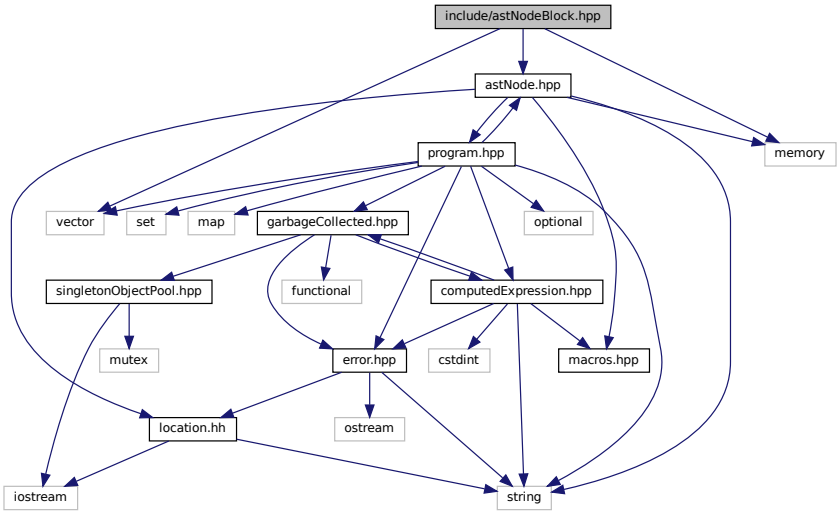
Declare the Tang::AstNodeBinary class.

6.6 include/astNodeBlock.hpp File Reference

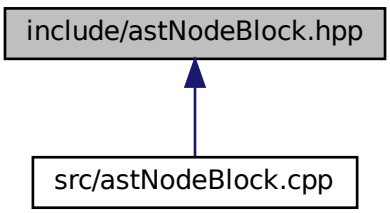
Declare the Tang::AstNodeBlock class.

```
#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.

6.6.1 Detailed Description

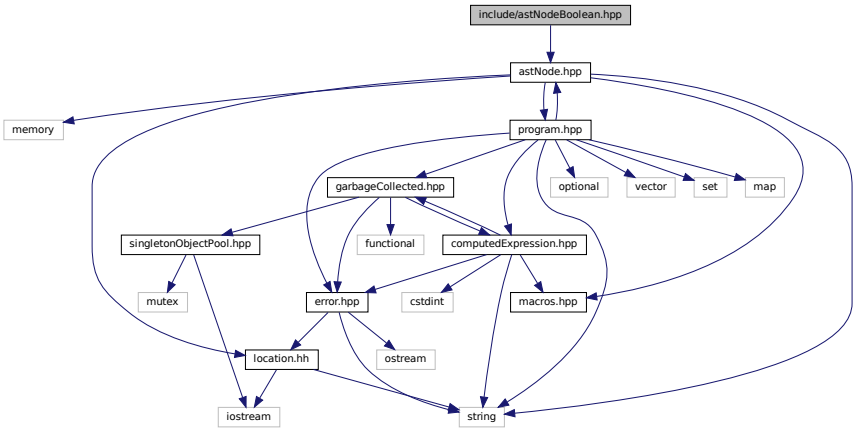
Declare the Tang::AstNodeBlock class.

6.7 include/astNodeBoolean.hpp File Reference

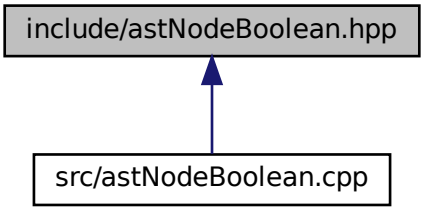
Declare the Tang::AstNodeBoolean class.

```
#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.

6.7.1 Detailed Description

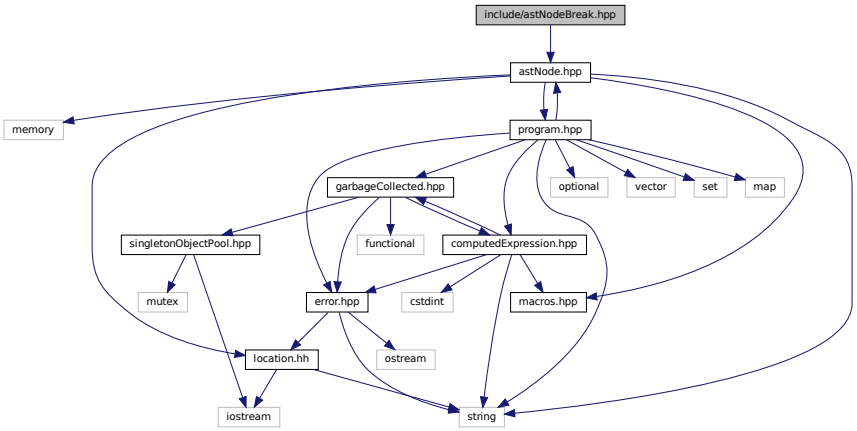
Declare the Tang::AstNodeBoolean class.

6.8 include/astNodeBreak.hpp File Reference

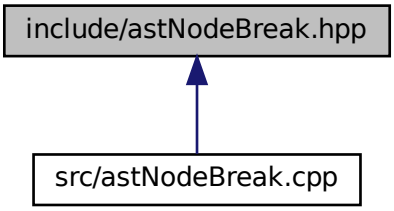
Declare the Tang::AstNodeBreak class.

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

Include dependency graph for astNodeBreak.hpp:



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



Classes

- class Tang::AstNodeBreak

  An AstNode that represents a `break` statement.

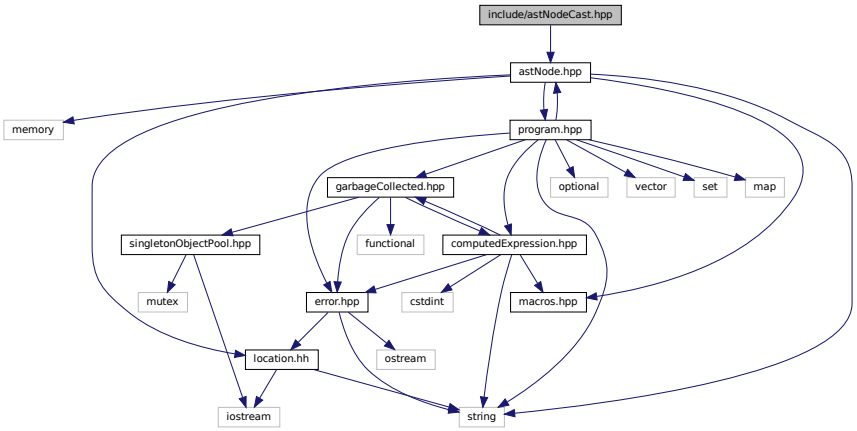
6.8.1 Detailed Description

Declare the Tang::AstNodeBreak class.

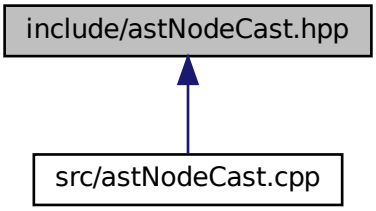
6.9 include/astNodeCast.hpp File Reference

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.

6.9.1 Detailed Description

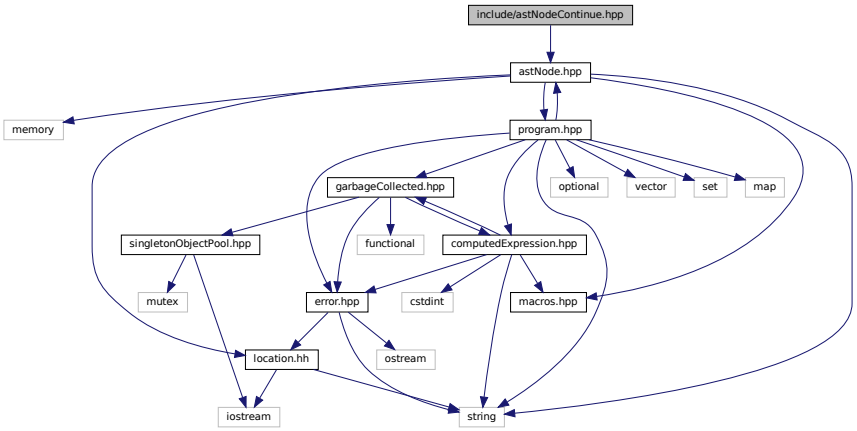
Declare the Tang::AstNodeCast class.

6.10 include/astNodeContinue.hpp File Reference

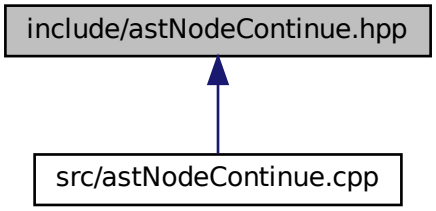
Declare the Tang::AstNodeContinue class.

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

Include dependency graph for astNodeContinue.hpp:



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



Classes

- class Tang::AstNodeContinue

  An AstNode that represents a `continue` statement.

6.10.1 Detailed Description

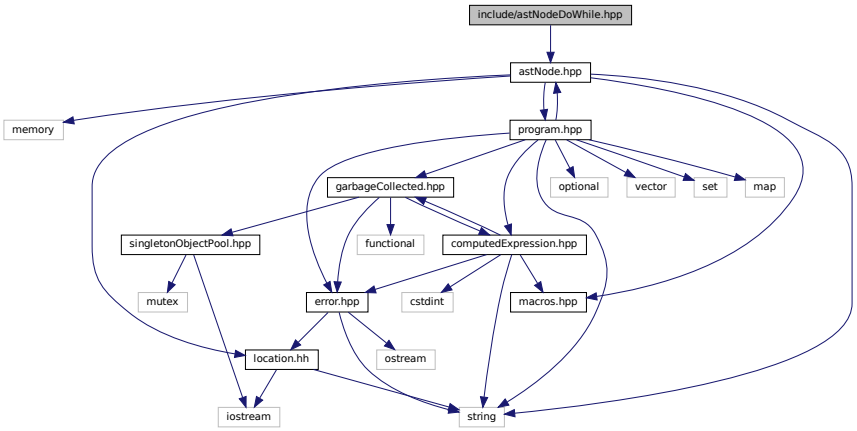
Declare the Tang::AstNodeContinue class.

6.11 include/astNodeDoWhile.hpp File Reference

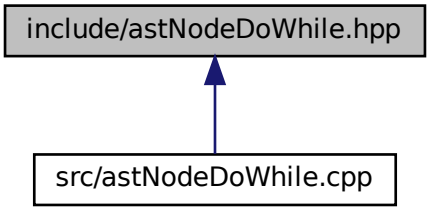
Declare the Tang::AstNodeDoWhile class.

```
#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.

6.11.1 Detailed Description

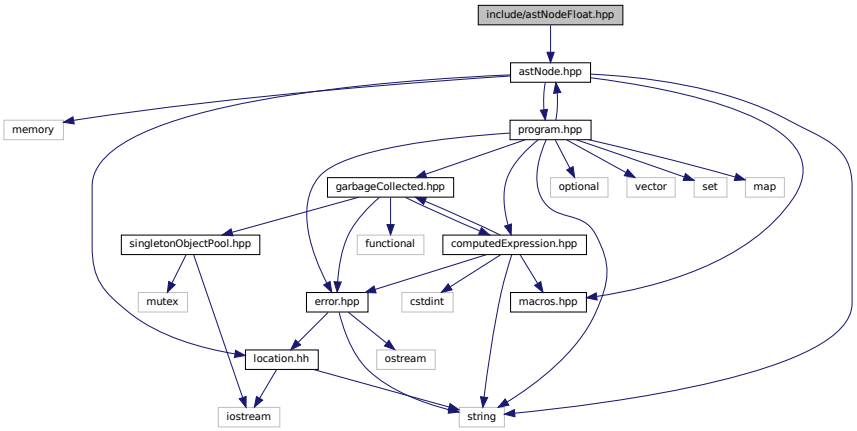
Declare the Tang::AstNodeDoWhile class.

6.12 include/astNodeFloat.hpp File Reference

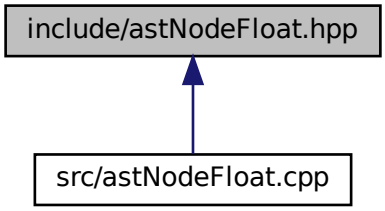
Declare the Tang::AstNodeFloat class.

```
#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.

6.12.1 Detailed Description

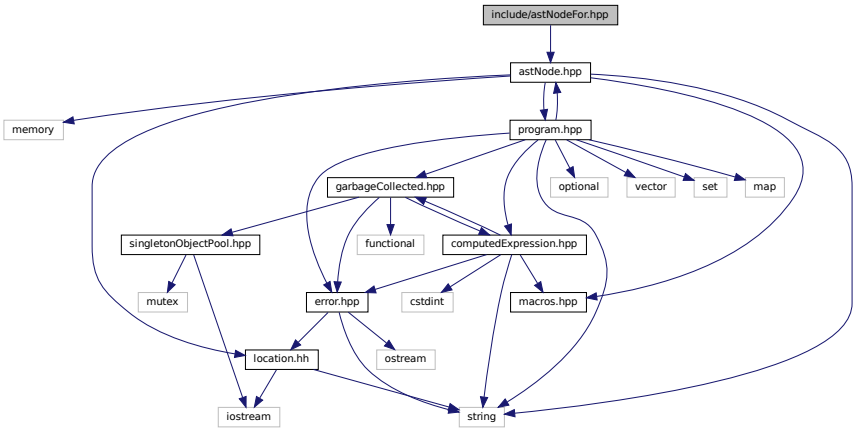
Declare the Tang::AstNodeFloat class.

6.13 include/astNodeFor.hpp File Reference

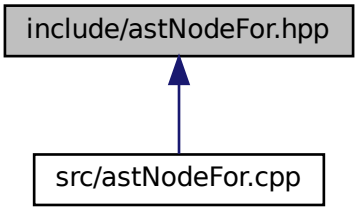
Declare the Tang::AstNodeFor class.

```
#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.

6.13.1 Detailed Description

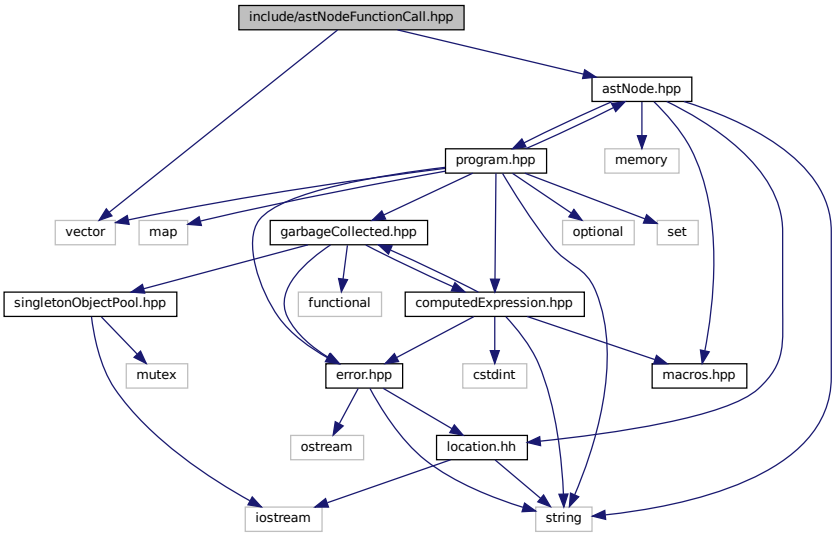
Declare the Tang::AstNodeFor class.

6.14 include/astNodeFunctionCall.hpp File Reference

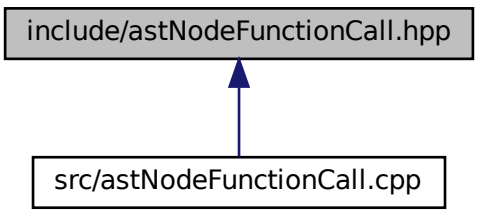
Declare the Tang::AstNodeFunctionCall class.

```
#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.

6.14.1 Detailed Description

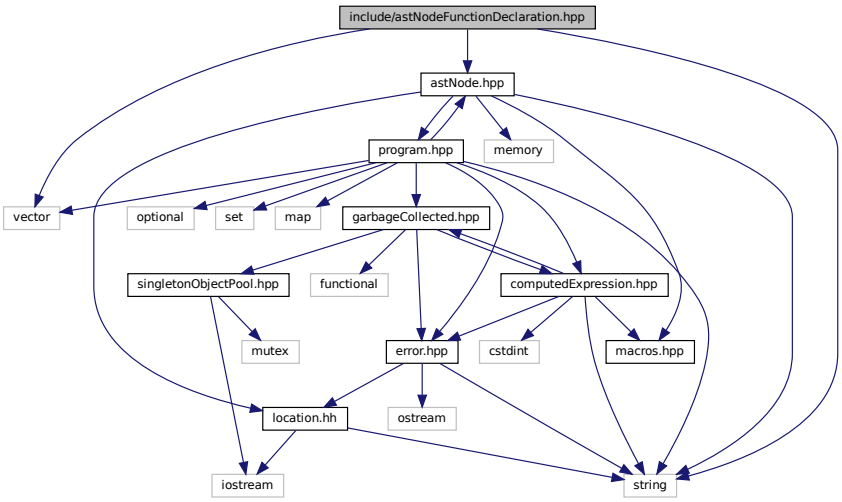
Declare the Tang::AstNodeFunctionCall class.

6.15 include/astNodeFunctionDeclaration.hpp File Reference

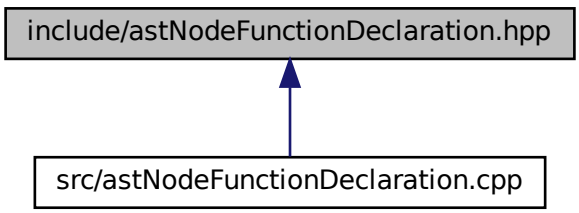
Declare the Tang::AstNodeFunctionDeclaration class.

```
#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.

6.15.1 Detailed Description

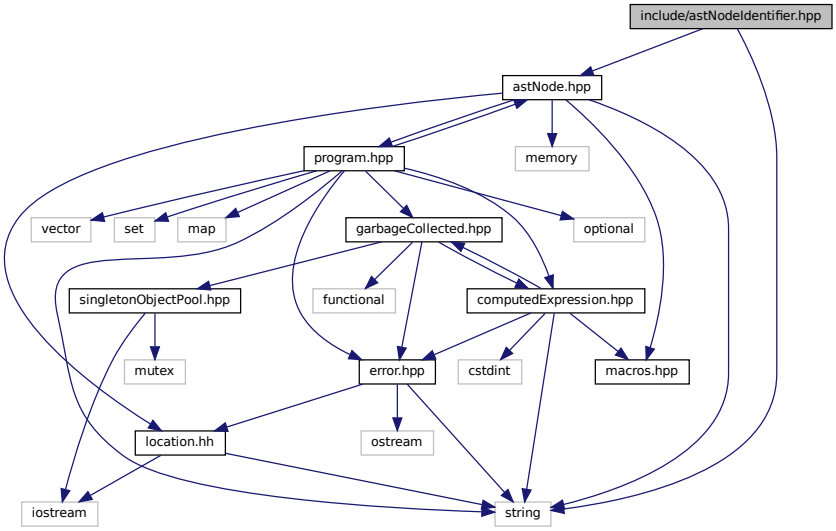
Declare the Tang::AstNodeFunctionDeclaration class.

6.16 include/astNodeIdentifier.hpp File Reference

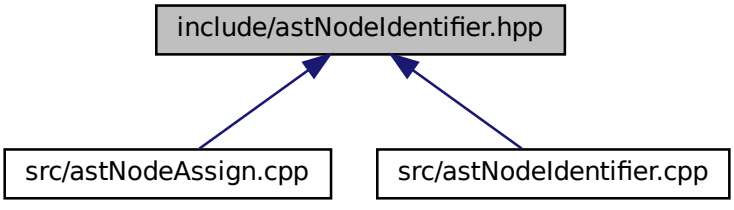
Declare the Tang::AstNodeIdentifier class.

```
#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.

6.16.1 Detailed Description

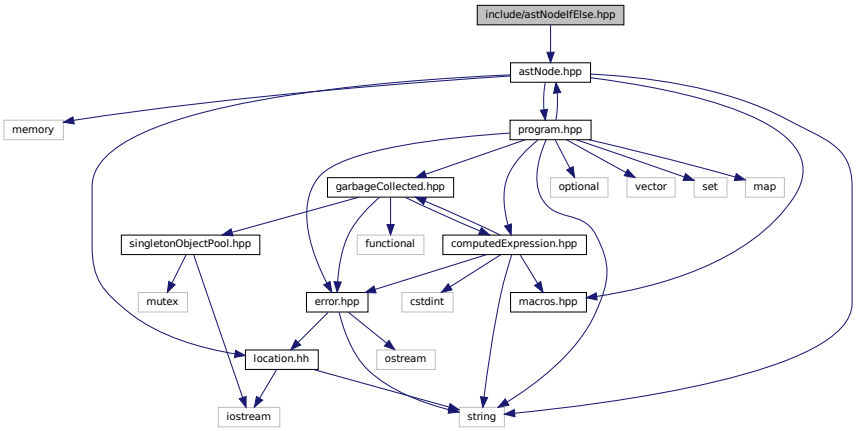
Declare the Tang::AstNodeIdentifier class.

6.17 include/astNodeIfElse.hpp File Reference

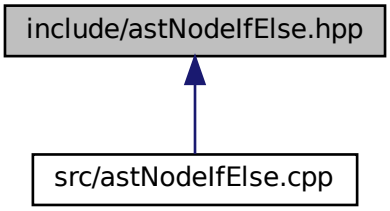
Declare the Tang::AstNodeIfElse class.

```
#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.

6.17.1 Detailed Description

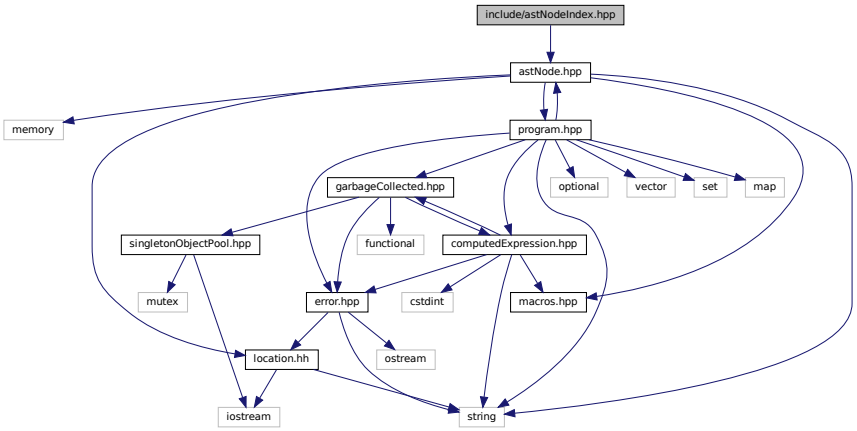
Declare the Tang::AstNodeIfElse class.

6.18 include/astNodeIndex.hpp File Reference

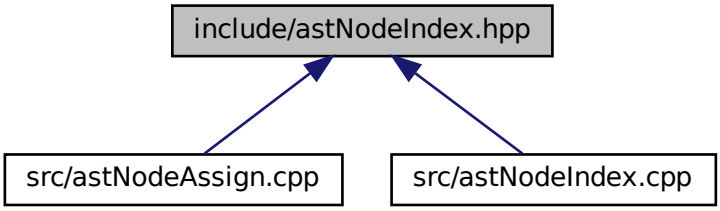
Declare the Tang::AstNodeIndex class.

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

Include dependency graph for astNodeIndex.hpp:



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



Classes

- class Tang::AstNodeIndex

  An AstNode that represents an index into a collection.

6.18.1 Detailed Description

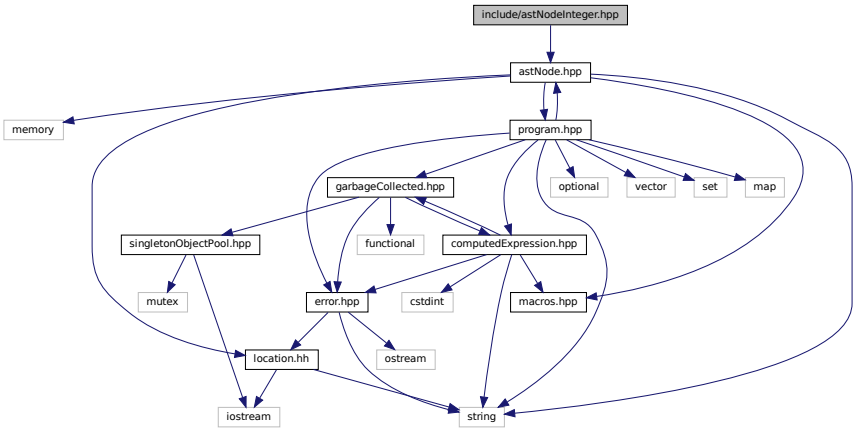
Declare the Tang::AstNodeIndex class.

6.19 include/astNodeInteger.hpp File Reference

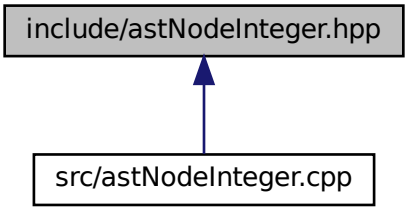
Declare the Tang::AstNodeInteger class.

```
#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.

6.19.1 Detailed Description

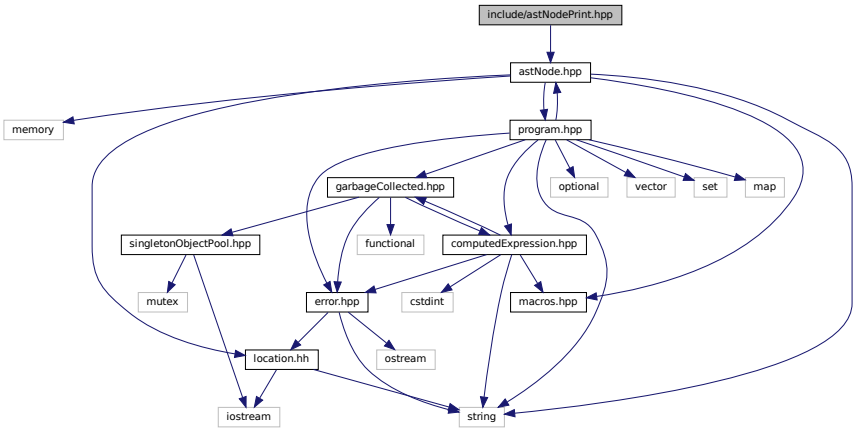
Declare the Tang::AstNodeInteger class.

6.20 include/astNodePrint.hpp File Reference

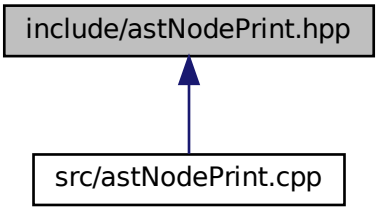
Declare the Tang::AstNodePrint class.

```
#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.

6.20.1 Detailed Description

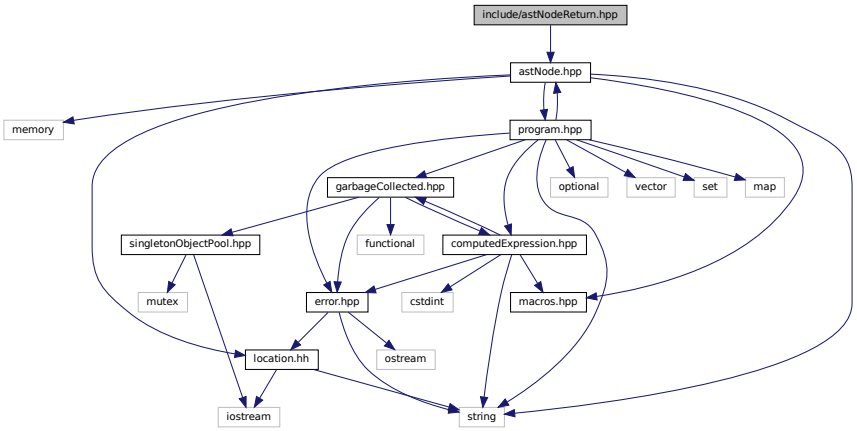
Declare the Tang::AstNodePrint class.

6.21 include/astNodeReturn.hpp File Reference

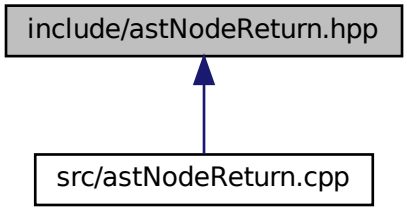
Declare the Tang::AstNodeReturn class.

```
#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.

6.21.1 Detailed Description

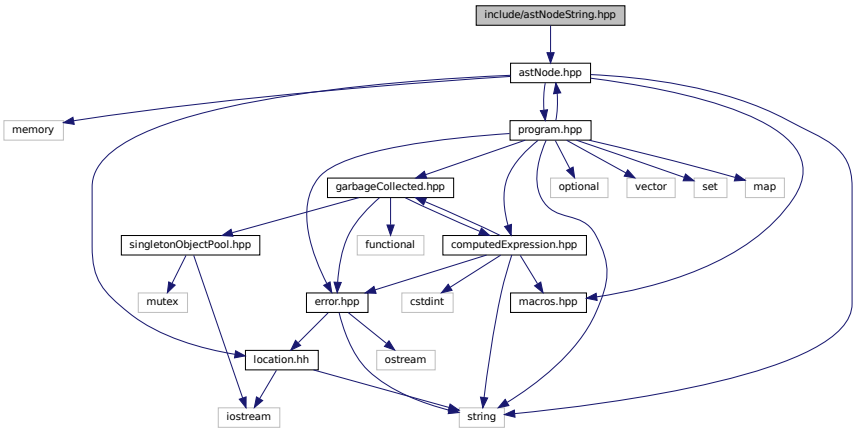
Declare the Tang::AstNodeReturn class.

6.22 include/astNodeString.hpp File Reference

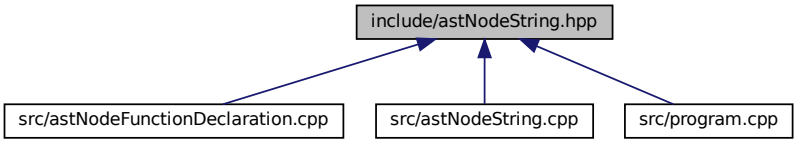
Declare the Tang::AstNodeString class.

```
#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.

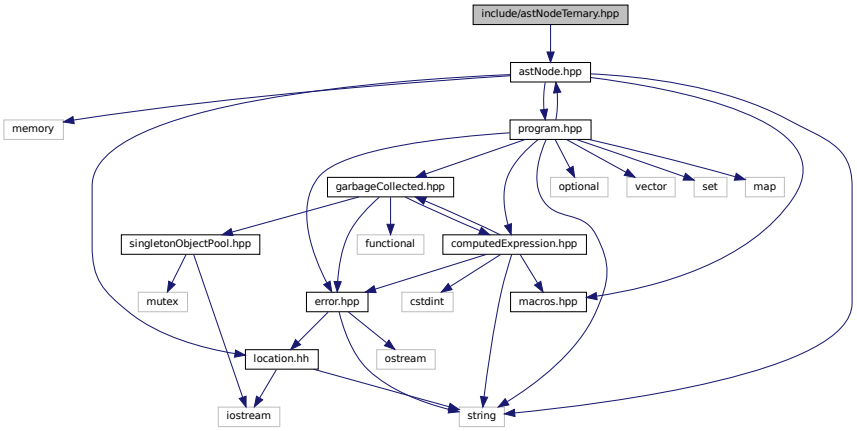
6.22.1 Detailed Description

Declare the Tang::AstNodeString class.

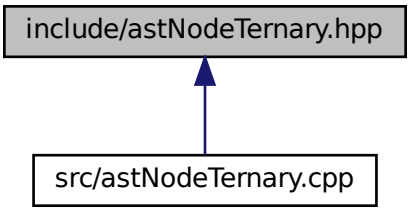
6.23 include/astNodeTernary.hpp File Reference

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.

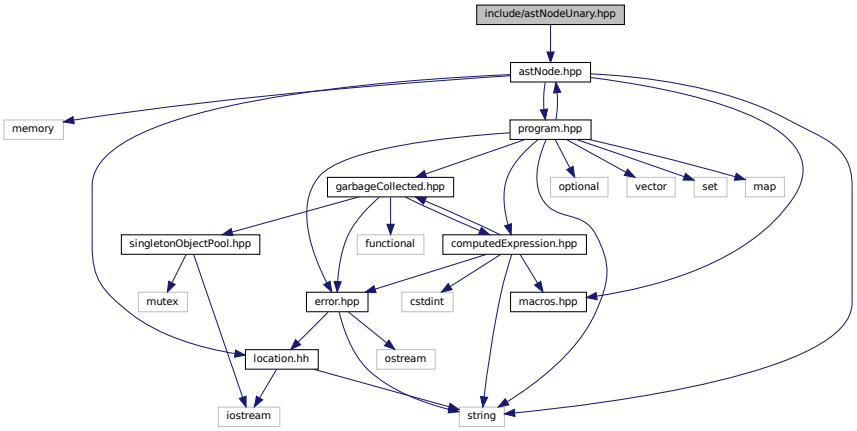
6.23.1 Detailed Description

Declare the Tang::AstNodeTernary class.

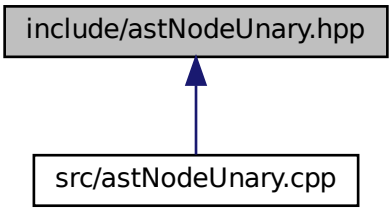
6.24 include/astNodeUnary.hpp File Reference

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.

6.24.1 Detailed Description

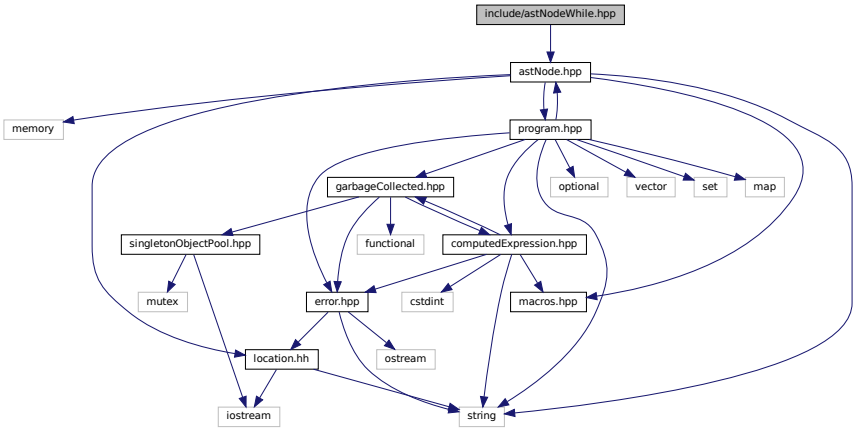
Declare the Tang::AstNodeUnary class.

6.25 include/astNodeWhile.hpp File Reference

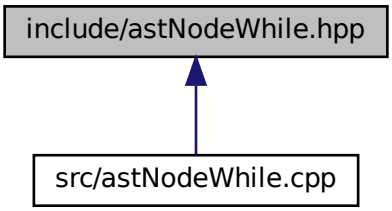
Declare the Tang::AstNodeWhile class.

```
#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.

6.25.1 Detailed Description

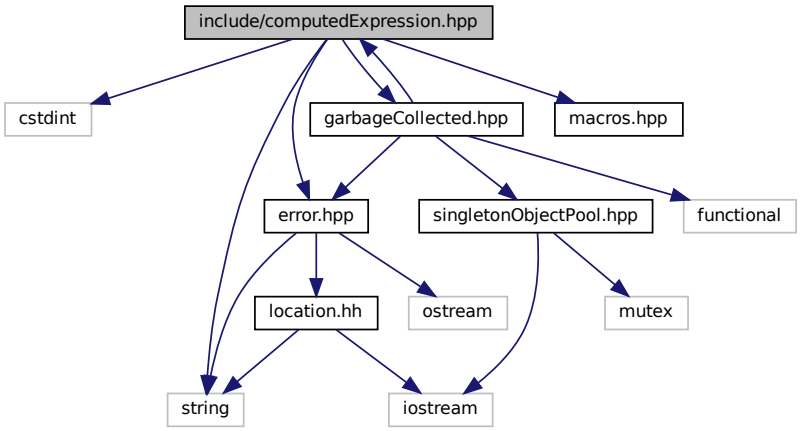
Declare the Tang::AstNodeWhile class.

6.26 include/computedExpression.hpp File Reference

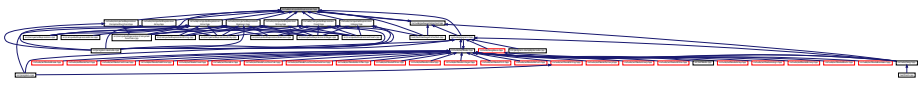
Declare the Tang::ComputedExpression base class.

```
#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.

6.26.1 Detailed Description

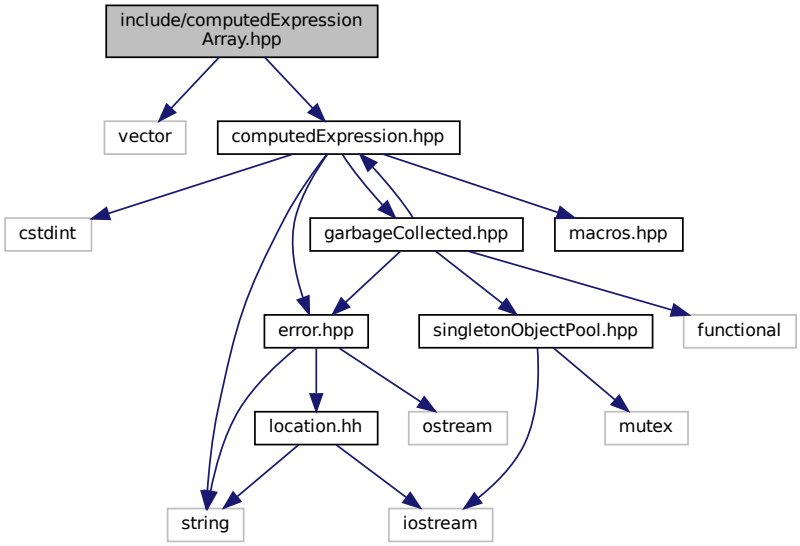
Declare the Tang::ComputedExpression base class.

6.27 include/computedExpressionArray.hpp File Reference

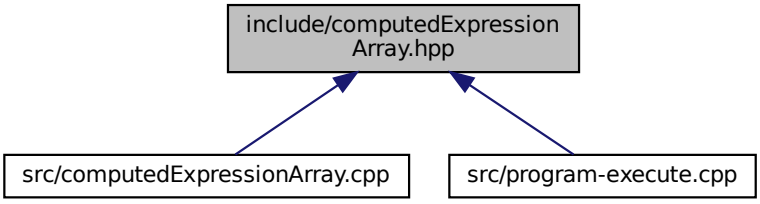
Declare the Tang::ComputedExpressionArray class.

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

Include dependency graph for computedExpressionArray.hpp:



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



Classes

- class Tang::ComputedExpressionArray

  Represents an Array that is the result of a computation.

6.27.1 Detailed Description

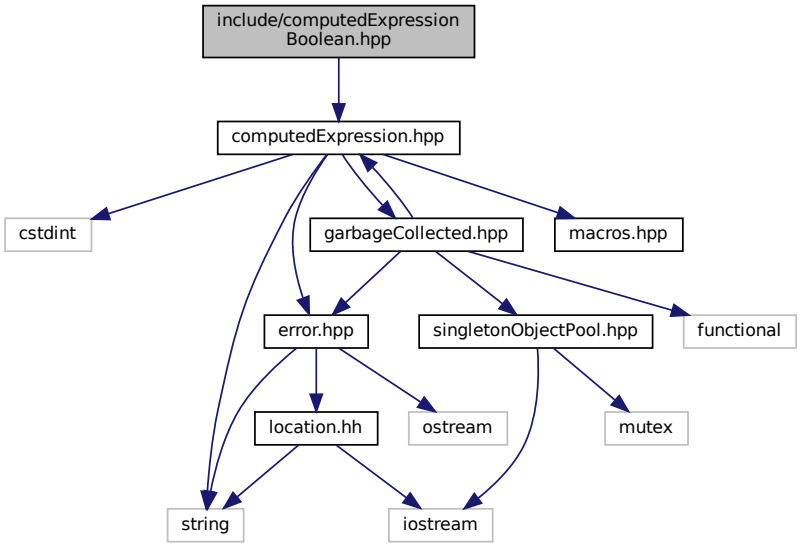
Declare the Tang::ComputedExpressionArray class.

6.28 include/computedExpressionBoolean.hpp File Reference

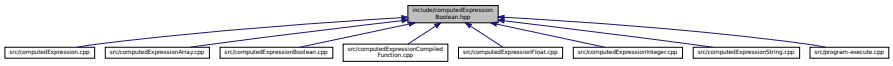
Declare the Tang::ComputedExpressionBoolean class.

```
#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.

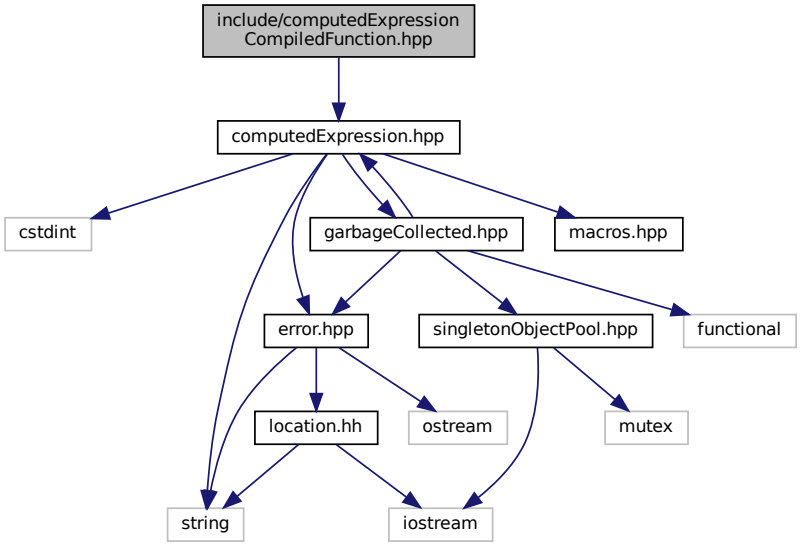
6.28.1 Detailed Description

Declare the Tang::ComputedExpressionBoolean class.

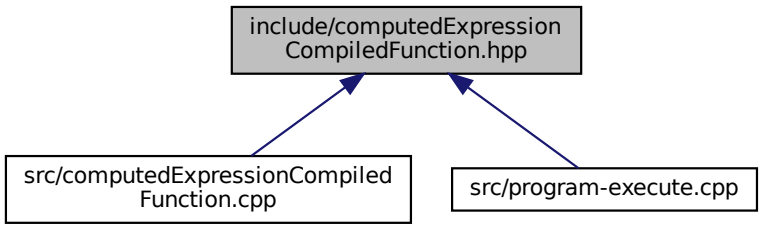
6.29 include/computedExpressionCompiledFunction.hpp File Reference

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.

6.29.1 Detailed Description

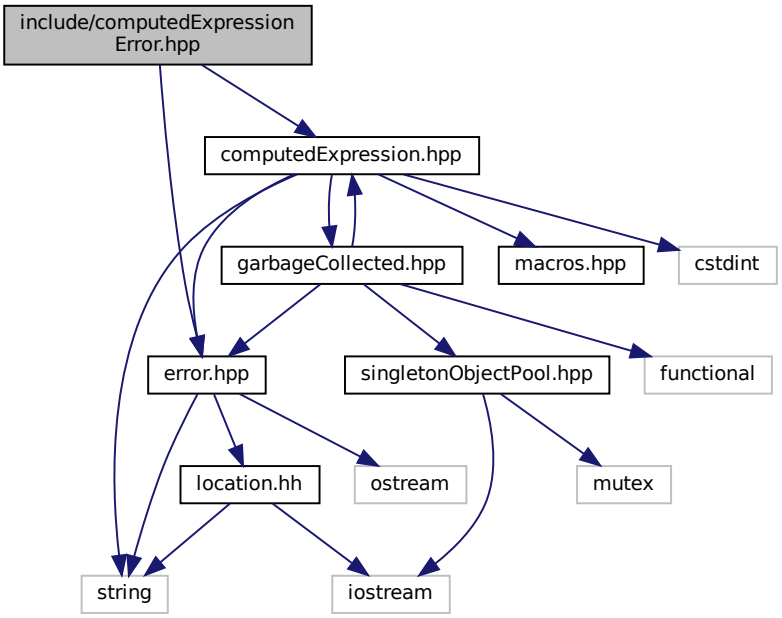
Declare the Tang::ComputedExpressionCompiledFunction class.

6.30 include/computedExpressionError.hpp File Reference

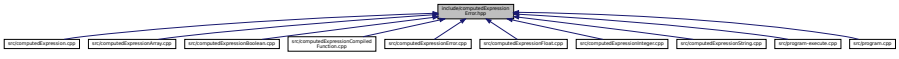
Declare the Tang::ComputedExpressionError class.

```
#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.

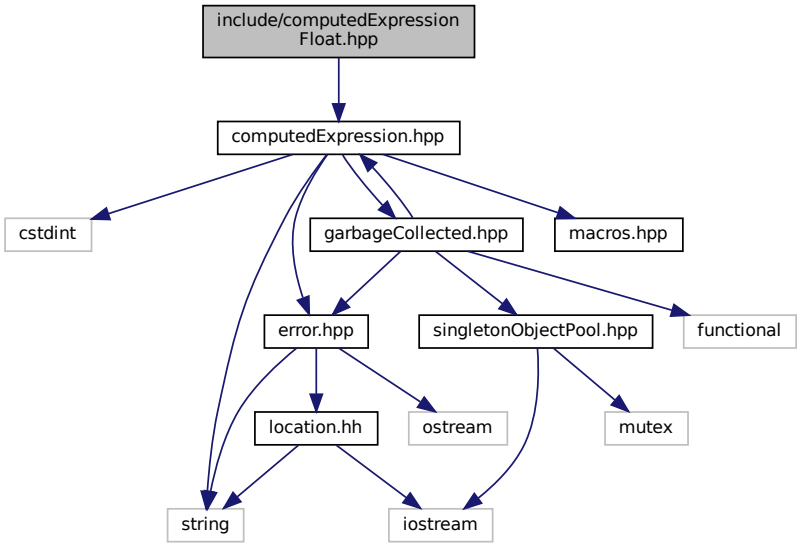
6.30.1 Detailed Description

Declare the Tang::ComputedExpressionError class.

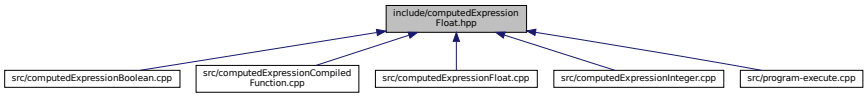
6.31 include/computedExpressionFloat.hpp File Reference

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.

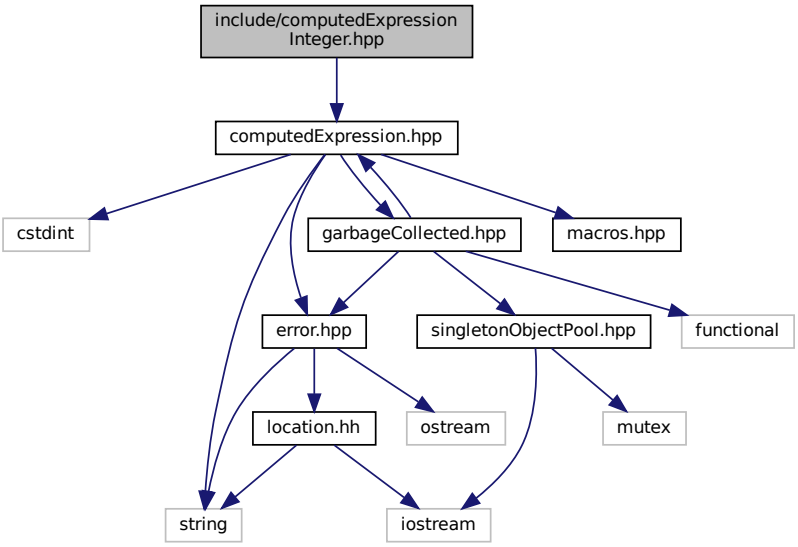
6.31.1 Detailed Description

Declare the Tang::ComputedExpressionFloat class.

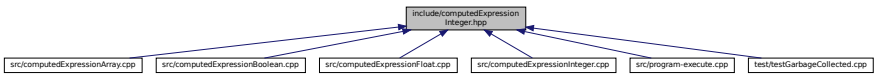
6.32 include/computedExpressionInteger.hpp File Reference

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.

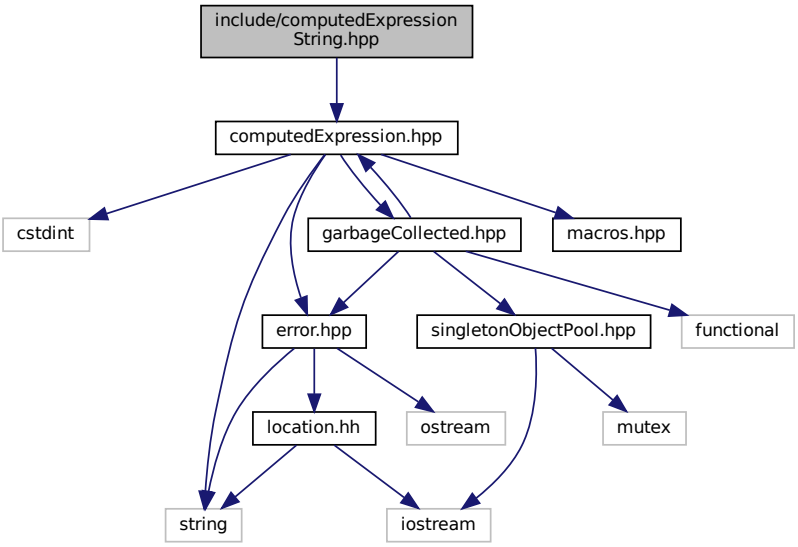
6.32.1 Detailed Description

Declare the Tang::ComputedExpressionInteger class.

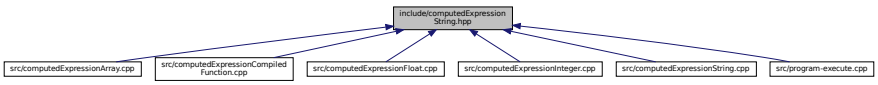
6.33 include/computedExpressionString.hpp File Reference

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.

6.33.1 Detailed Description

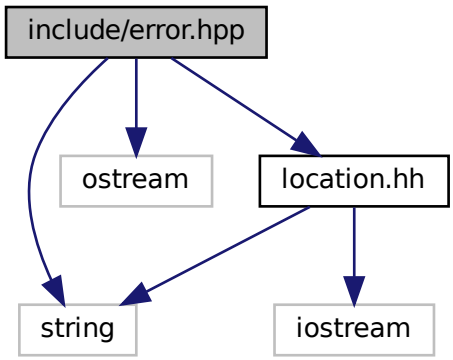
Declare the Tang::ComputedExpressionString class.

6.34 include/error.hpp File Reference

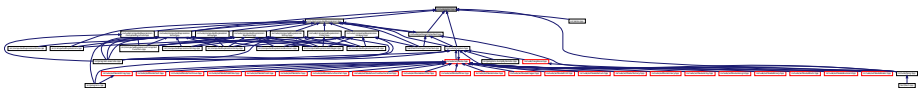
Declare the Tang::Error class used to describe syntax and runtime errors.

```
#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.

6.34.1 Detailed Description

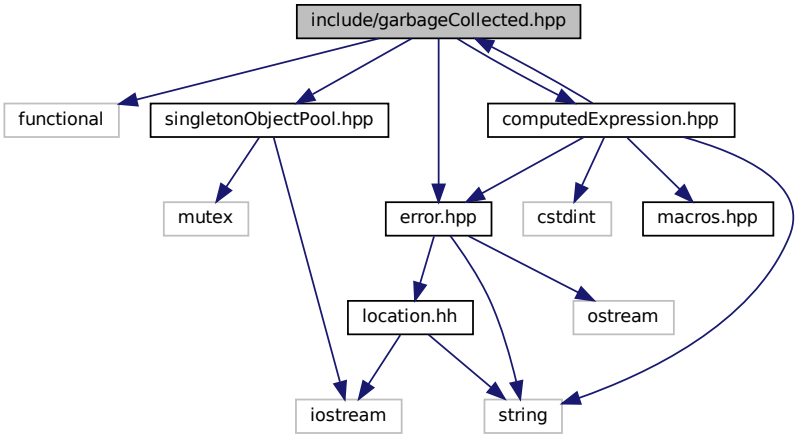
Declare the Tang::Error class used to describe syntax and runtime errors.

6.35 include/garbageCollected.hpp File Reference

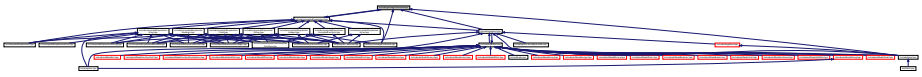
Declare the Tang::GarbageCollected class.

```
#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.

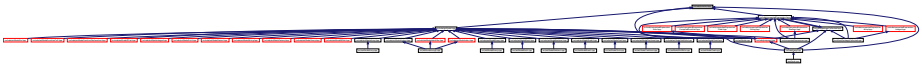
6.35.1 Detailed Description

Declare the Tang::GarbageCollected class.

6.36 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.

6.36.1 Detailed Description

Contains generic macros.

6.37 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 , COPY ,
  JMP , JMPF , JMPF_POP , JMPT ,
  JMPT_POP , NULLVAL , INTEGER , FLOAT ,
  BOOLEAN , STRING , ARRAY , FUNCTION ,
  ASSIGNINDEX , ADD , SUBTRACT , MULTIPLY ,
  DIVIDE , MODULO , NEGATIVE , NOT ,
  LT , LTE , GT , GTE ,
  EQ , NEQ , INDEX , CASTINTEGER ,
  CASTFLOAT , CASTBOOLEAN , CALLFUNC , RETURN ,
  PRINT }

6.37.1 Detailed Description

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

6.37.2 Enumeration Type Documentation

6.37.2.1 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 #.
COPY	Stack # (from fp): Deep copy val @ stack #, 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.
ARRAY	Get len, pop `len` items, putting them into an array with the last array item popped first.
FUNCTION	Get argc, PC#: push function(argc, PC #)
ASSIGNINDEX	Pop index, pop collection, pop value, push (collection[index] = value)
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.
INDEX	Pop index, pop collection, push collection[index].
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.

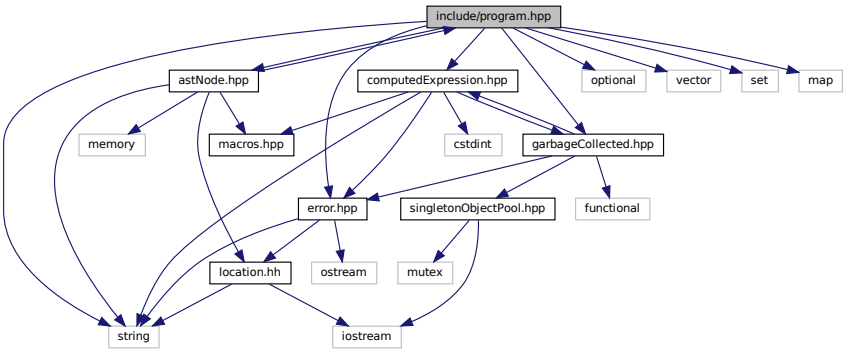
6.38 include/program.hpp File Reference

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

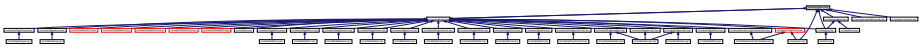
```
#include <string>
#include <optional>
#include <vector>
```

```
#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.

6.38.1 Detailed Description

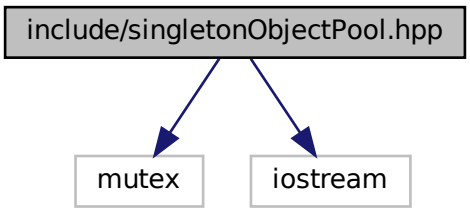
Declare the Tang::Program class used to compile and execute source code.

6.39 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.

6.39.1 Detailed Description

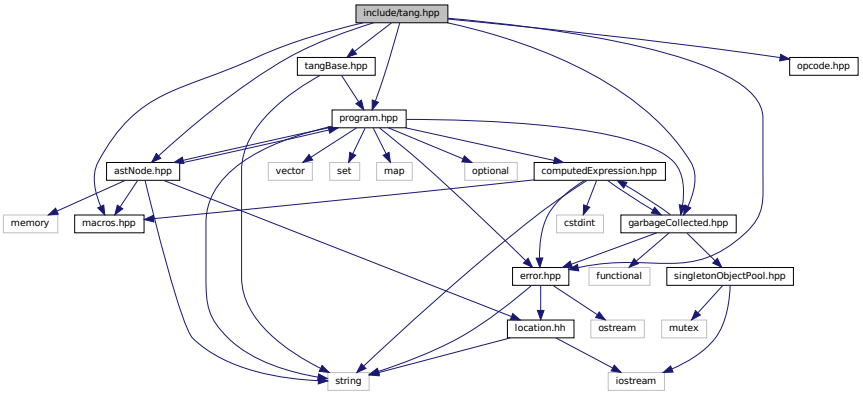
Declare the Tang::SingletonObjectPool class.

6.40 include/tang.hpp File Reference

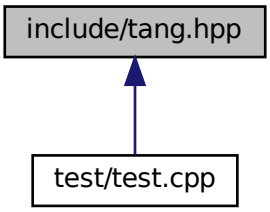
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

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

Include dependency graph for tang.hpp:



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



6.40.1 Detailed Description

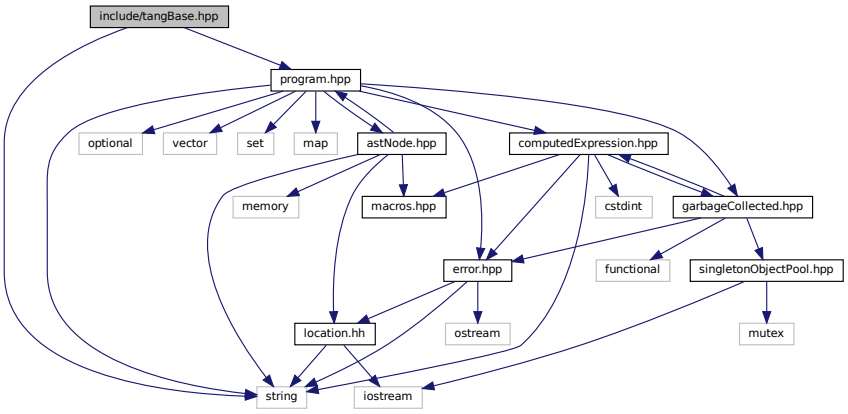
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

6.41 include/tangBase.hpp File Reference

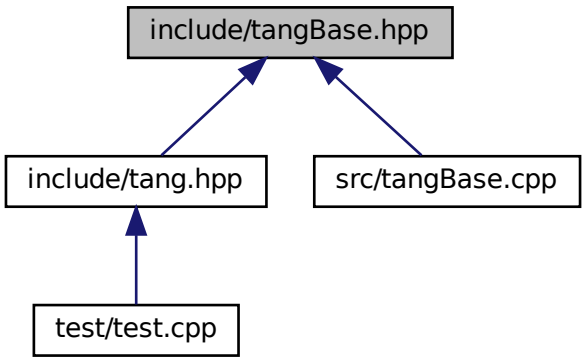
Declare the Tang::TangBase class used to interact with Tang.

```
#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.

6.41.1 Detailed Description

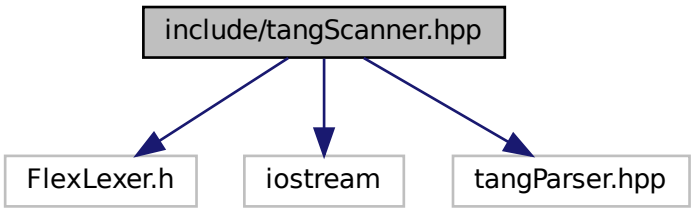
Declare the Tang::TangBase class used to interact with Tang.

6.42 include/tangScanner.hpp File Reference

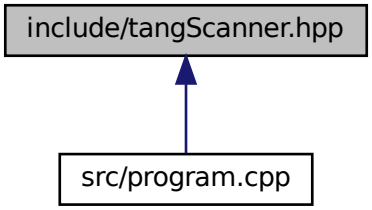
Declare the Tang::TangScanner used to tokenize a Tang script.

```
#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()

6.42.1 Detailed Description

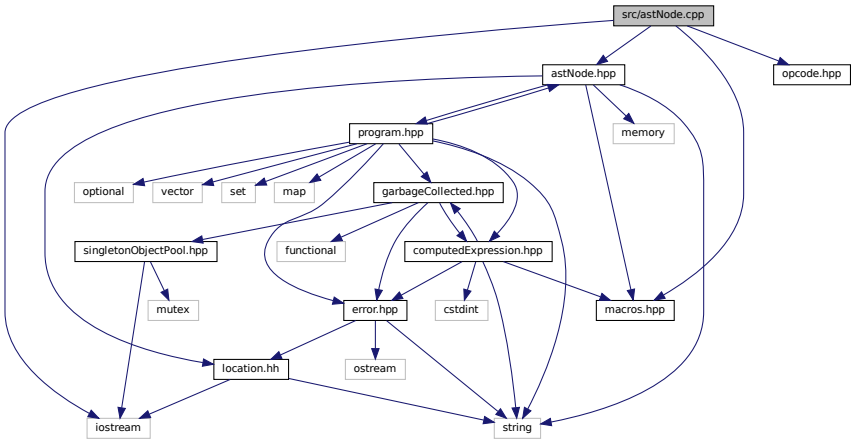
Declare the Tang::TangScanner used to tokenize a Tang script.

6.43 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:



6.43.1 Detailed Description

Define the Tang::AstNode class.

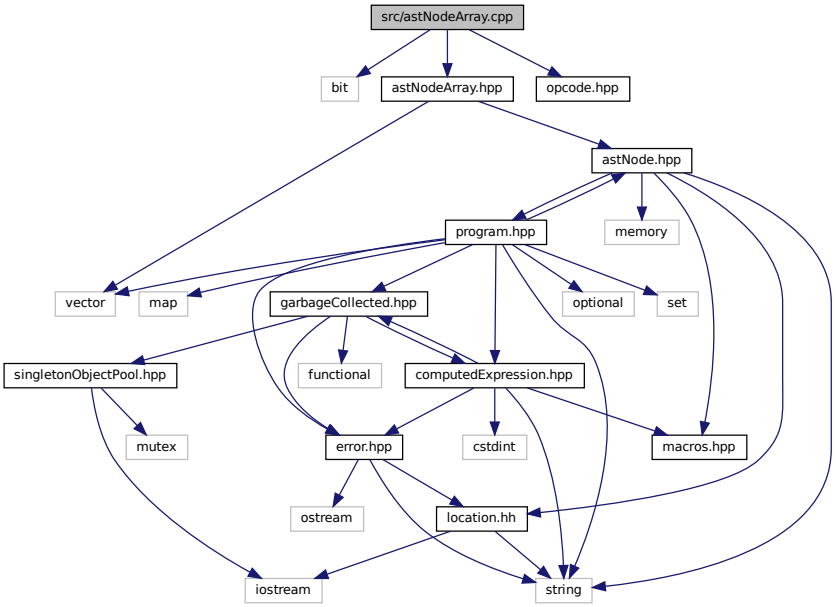
6.44 src/astNodeArray.cpp File Reference

Define the Tang::AstNodeArray class.

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

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

Include dependency graph for astNodeArray.cpp:



6.44.1 Detailed Description

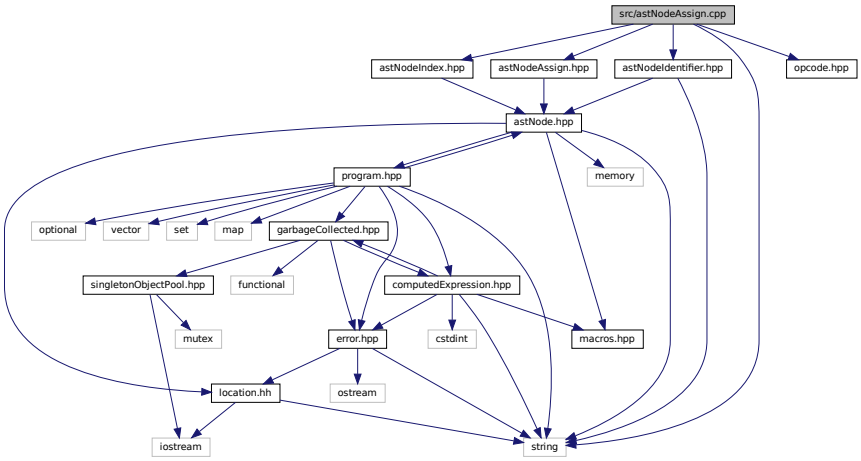
Define the Tang::AstNodeArray class.

6.45 src/astNodeAssign.cpp File Reference

Define the Tang::AstNodeAssign class.

```
#include <string>
#include "astNodeAssign.hpp"
#include "astNodeIdentifier.hpp"
#include "astNodeIndex.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeAssign.cpp:



6.45.1 Detailed Description

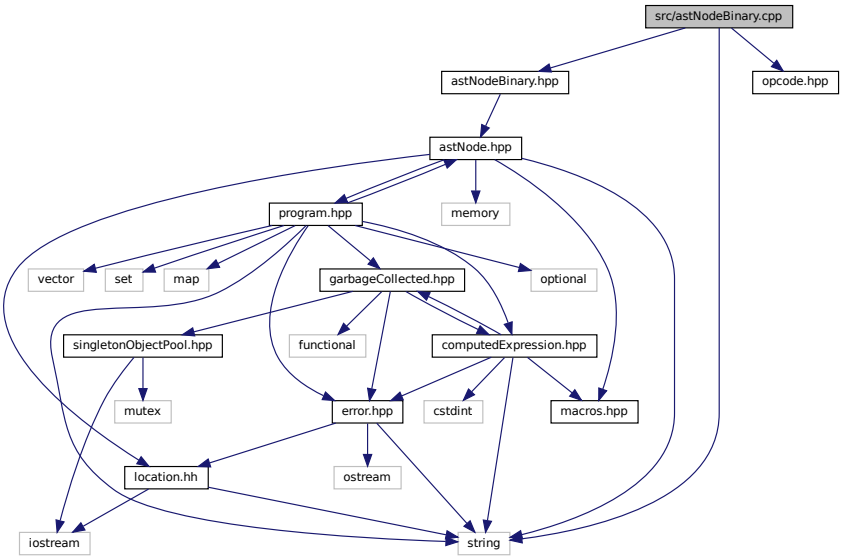
Define the Tang::AstNodeAssign class.

6.46 src/astNodeBinary.cpp File Reference

Define the Tang::AstNodeBinary class.

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

Include dependency graph for astNodeBinary.cpp:



6.46.1 Detailed Description

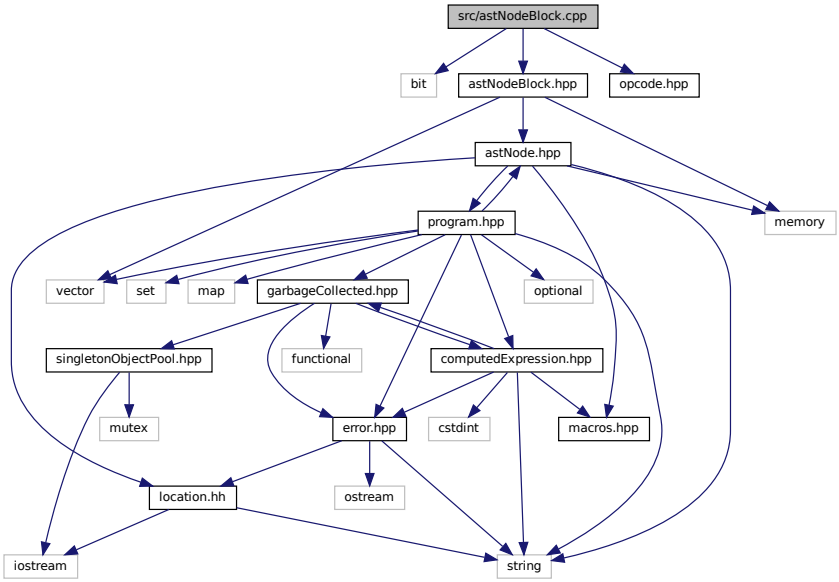
Define the Tang::AstNodeBinary class.

6.47 src/astNodeBlock.cpp File Reference

Define the Tang::AstNodeBlock class.

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

Include dependency graph for astNodeBlock.cpp:



6.47.1 Detailed Description

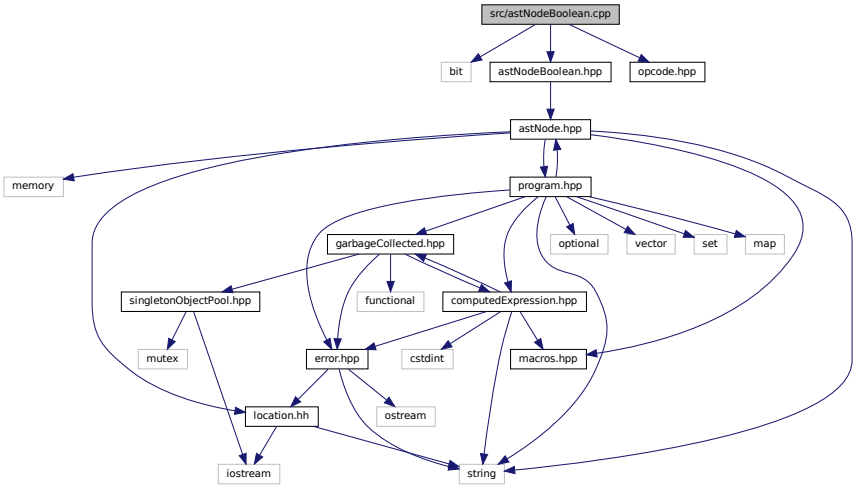
Define the Tang::AstNodeBlock class.

6.48 src/astNodeBoolean.cpp File Reference

Define the Tang::AstNodeBoolean class.

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

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



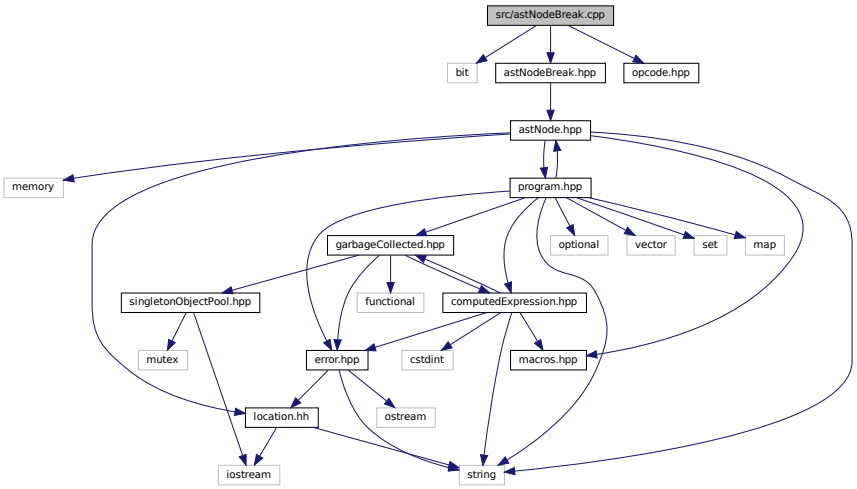
6.48.1 Detailed Description

Define the Tang::AstNodeBoolean class.

6.49 src/astNodeBreak.cpp File Reference

Define the Tang::AstNodeBreak class.

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



6.49.1 Detailed Description

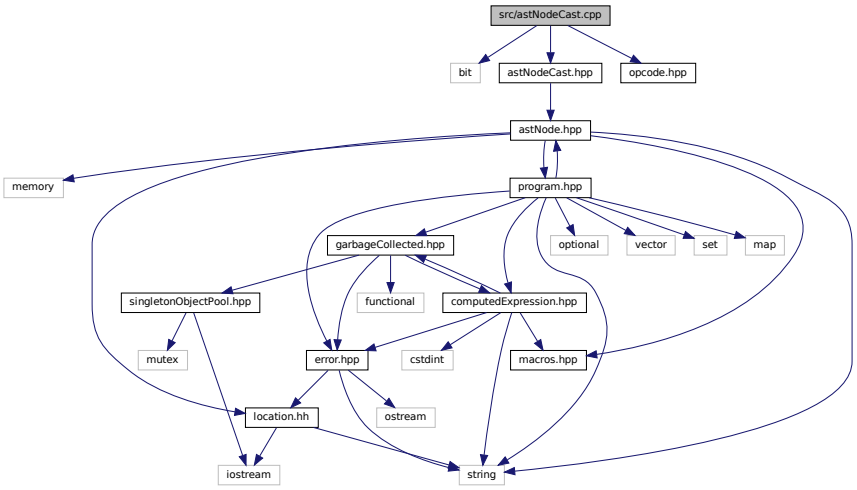
Define the Tang::AstNodeBreak class.

6.50 src/astNodeCast.cpp File Reference

Define the Tang::AstNodeCast class.

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

Include dependency graph for astNodeCast.cpp:



6.50.1 Detailed Description

Define the Tang::AstNodeCast class.

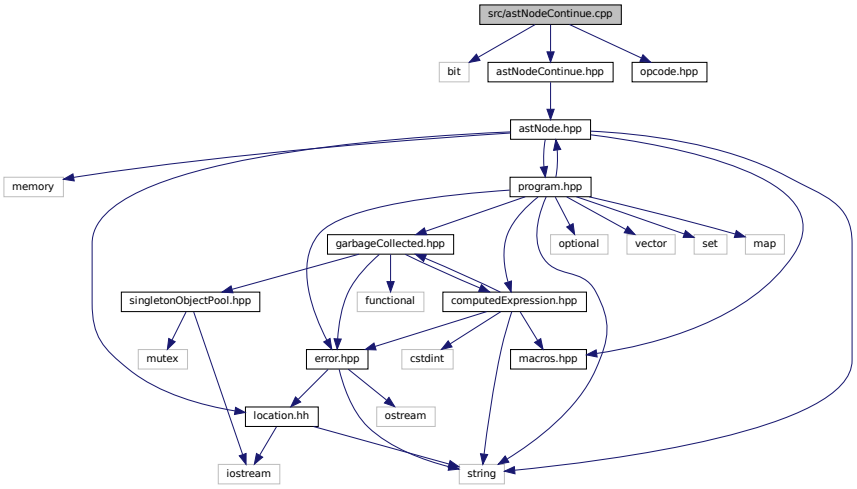
6.51 src/astNodeContinue.cpp File Reference

Define the Tang::AstNodeContinue class.

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

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

Include dependency graph for astNodeContinue.cpp:



6.51.1 Detailed Description

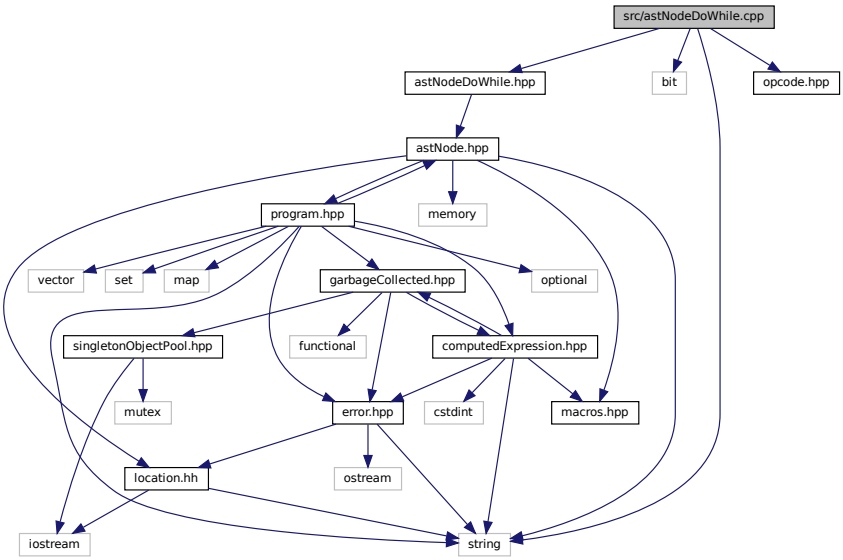
Define the Tang::AstNodeContinue class.

6.52 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:



6.52.1 Detailed Description

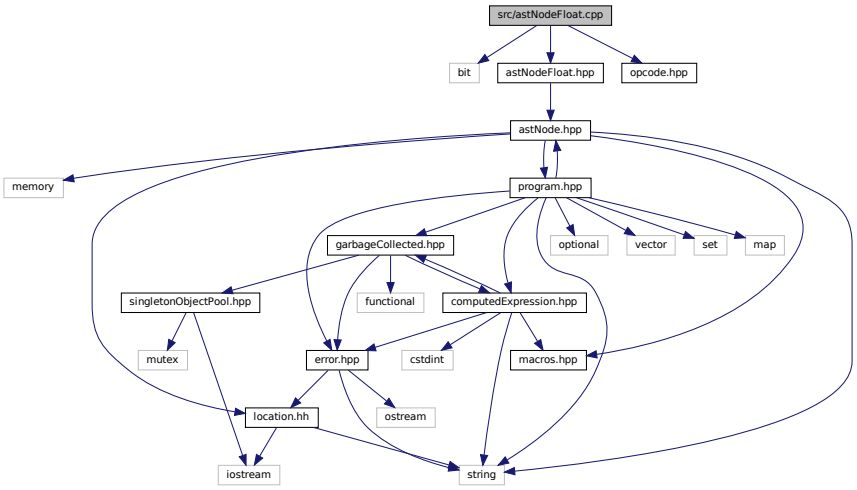
Define the Tang::AstNodeDoWhile class.

6.53 src/astNodeFloat.cpp File Reference

Define the Tang::AstNodeFloat class.

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

Include dependency graph for astNodeFloat.cpp:



6.53.1 Detailed Description

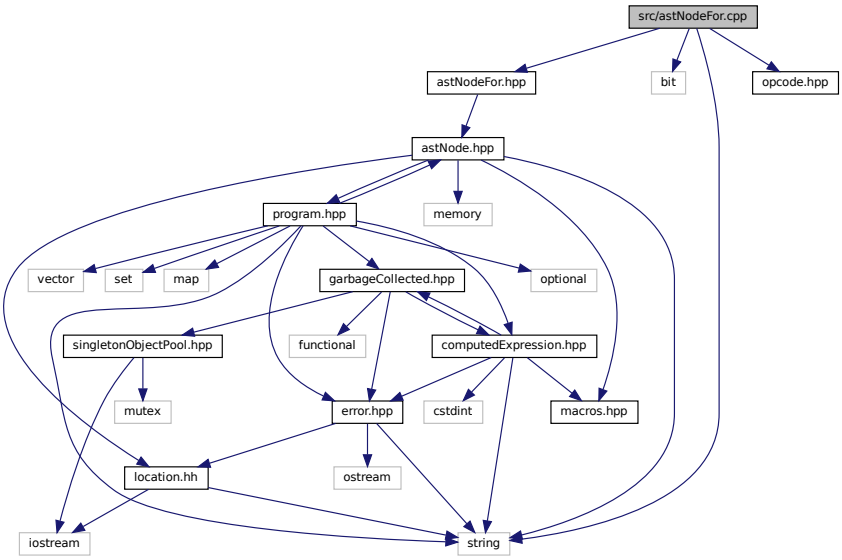
Define the Tang::AstNodeFloat class.

6.54 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:



6.54.1 Detailed Description

Define the Tang::AstNodeFor class.

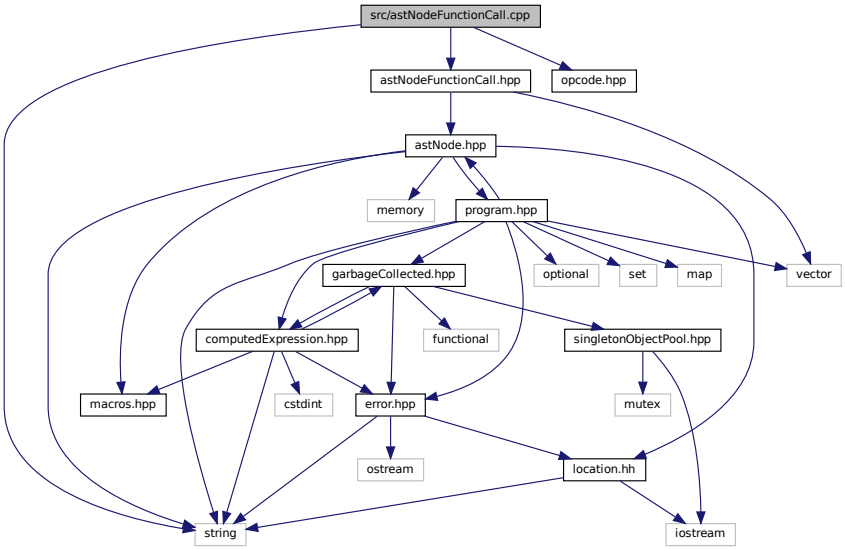
6.55 src/astNodeFunctionCall.cpp File Reference

Define the Tang::AstNodeFunctionCall class.

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

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

Include dependency graph for astNodeFunctionCall.cpp:



6.55.1 Detailed Description

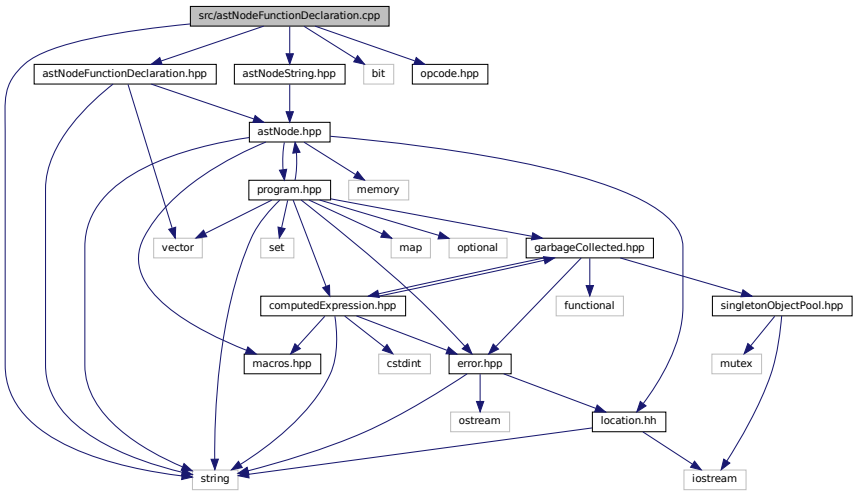
Define the Tang::AstNodeFunctionCall class.

6.56 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:



6.56.1 Detailed Description

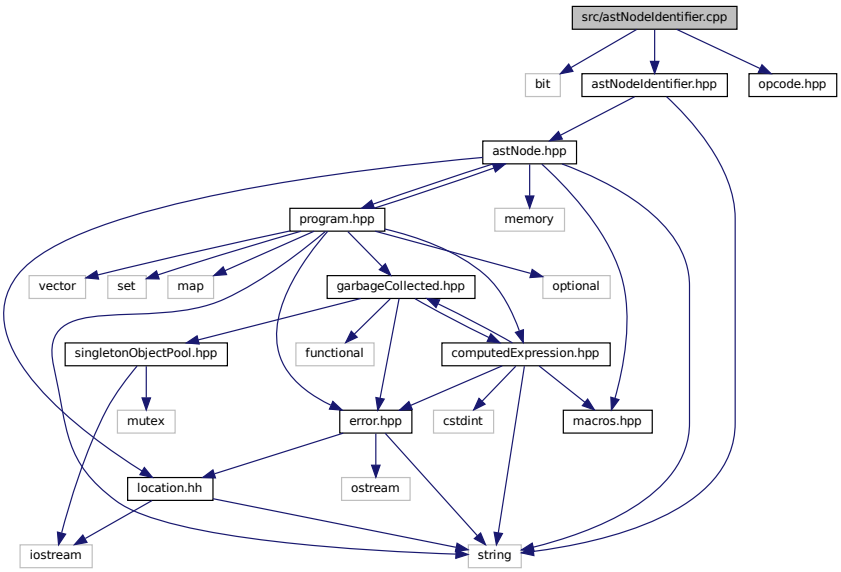
Define the Tang::AstNodeFunctionDeclaration class.

6.57 src/astNodeIdentifier.cpp File Reference

Define the Tang::AstNodeIdentifier class.

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

Include dependency graph for astNodeIdentifier.cpp:



6.57.1 Detailed Description

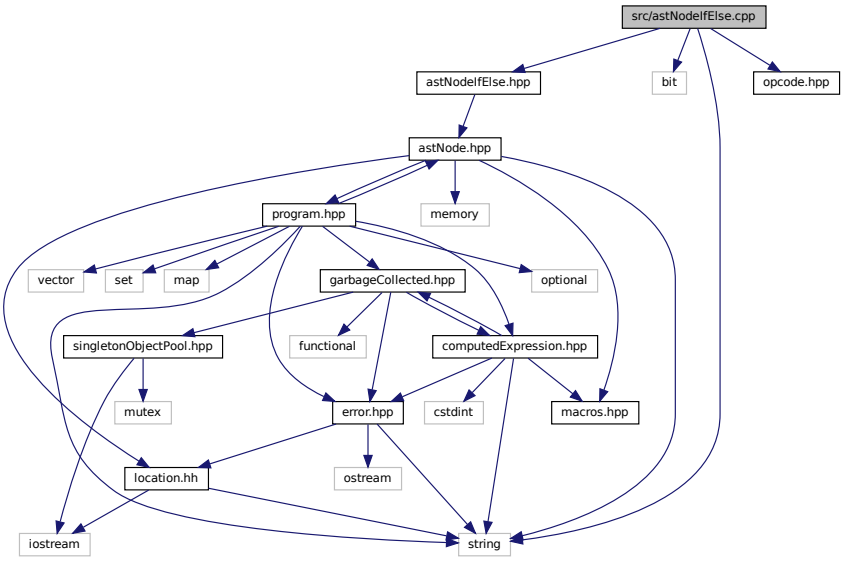
Define the Tang::AstNodeIdentifier class.

6.58 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:



6.58.1 Detailed Description

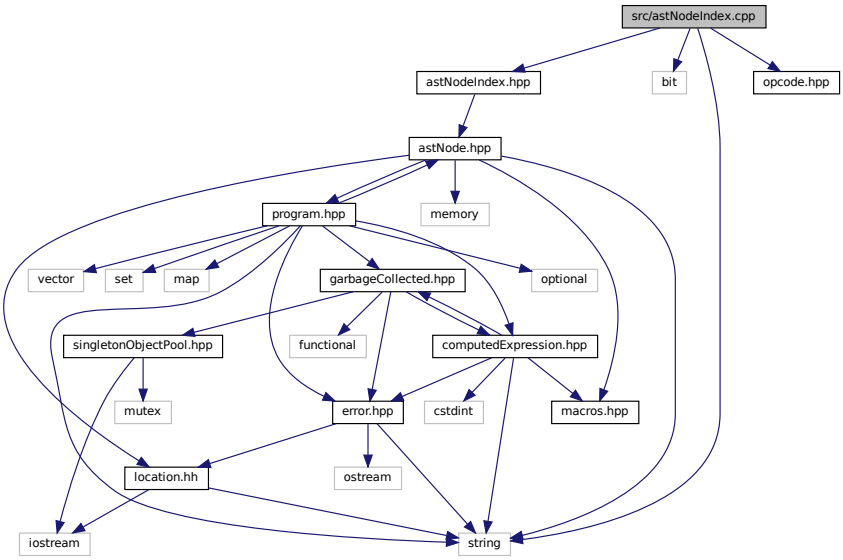
Define the Tang::AstNodeIfElse class.

6.59 src/astNodeIndex.cpp File Reference

Define the Tang::AstNodeIndex class.

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

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



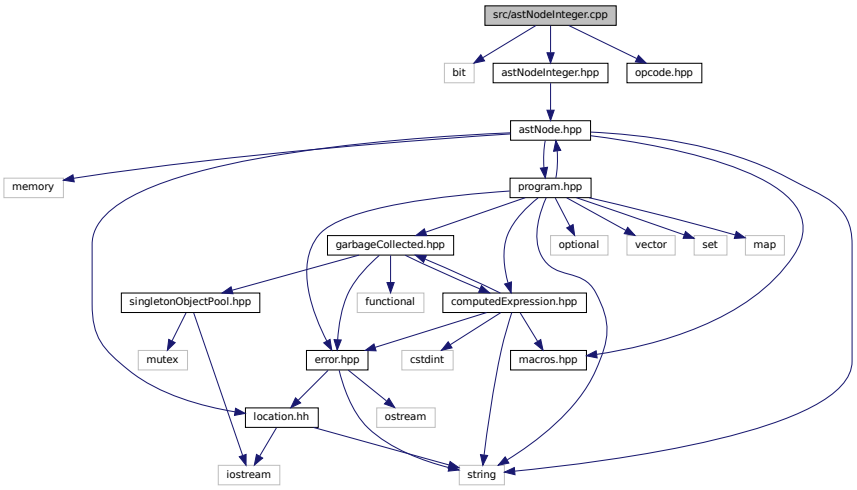
6.59.1 Detailed Description

Define the Tang::AstNodeIndex class.

6.60 src/astNodeInteger.cpp File Reference

Define the Tang::AstNodeInteger class.

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



6.60.1 Detailed Description

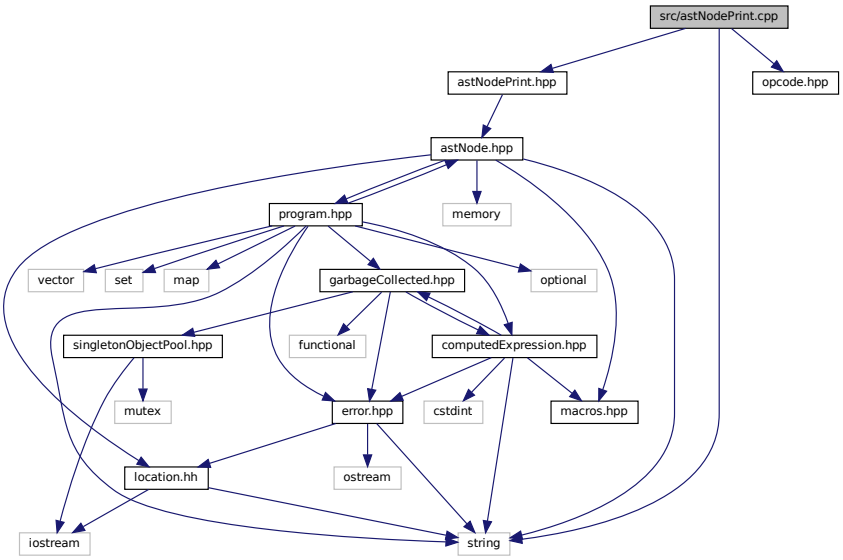
Define the Tang::AstNodeInteger class.

6.61 src/astNodePrint.cpp File Reference

Define the Tang::AstNodePrint class.

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

Include dependency graph for astNodePrint.cpp:



6.61.1 Detailed Description

Define the Tang::AstNodePrint class.

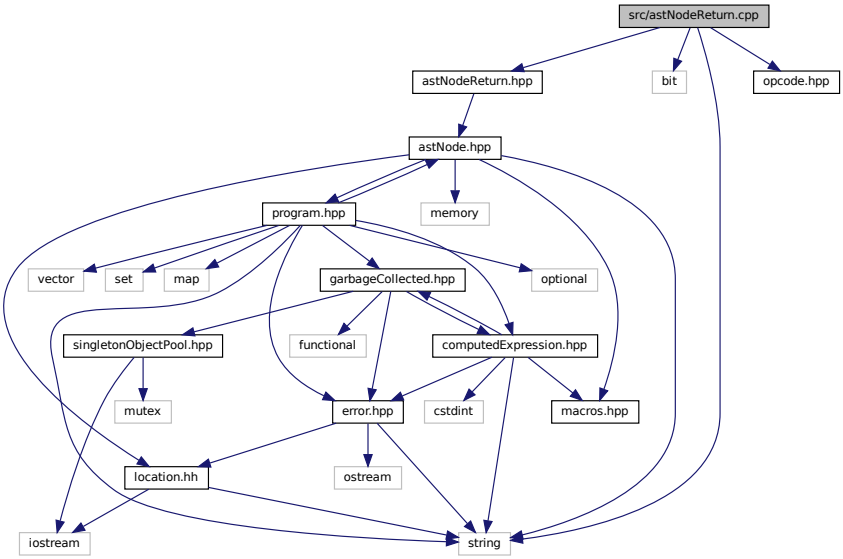
6.62 src/astNodeReturn.cpp File Reference

Define the Tang::AstNodeReturn class.

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

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

Include dependency graph for astNodeReturn.cpp:



6.62.1 Detailed Description

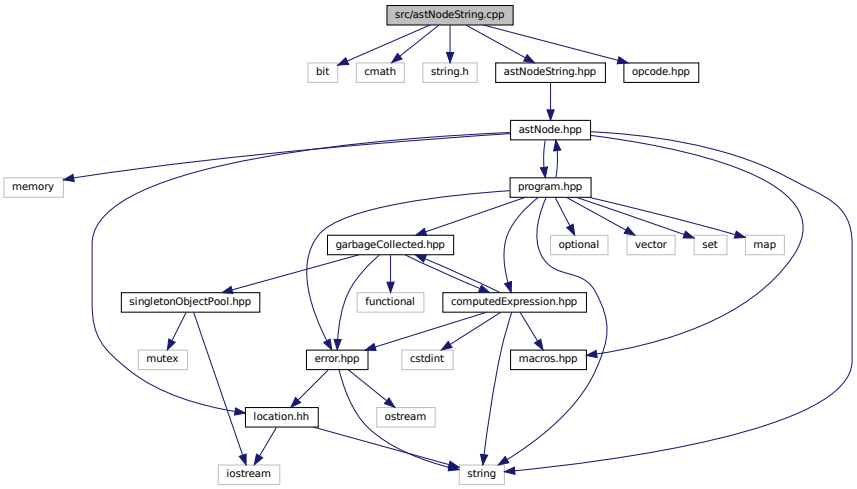
Define the Tang::AstNodeReturn class.

6.63 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:



6.63.1 Detailed Description

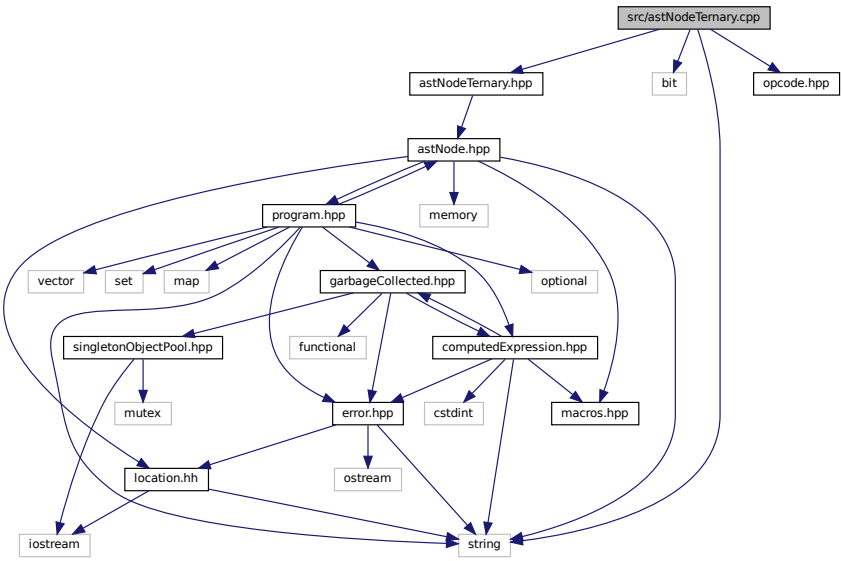
Define the Tang::AstNodeString class.

6.64 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:



6.64.1 Detailed Description

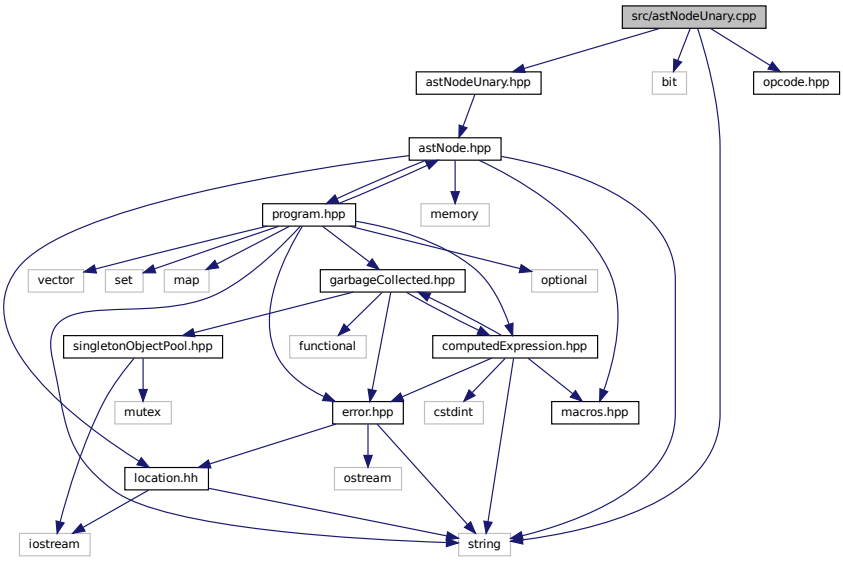
Define the Tang::AstNodeTernary class.

6.65 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:



6.65.1 Detailed Description

Define the Tang::AstNodeUnary class.

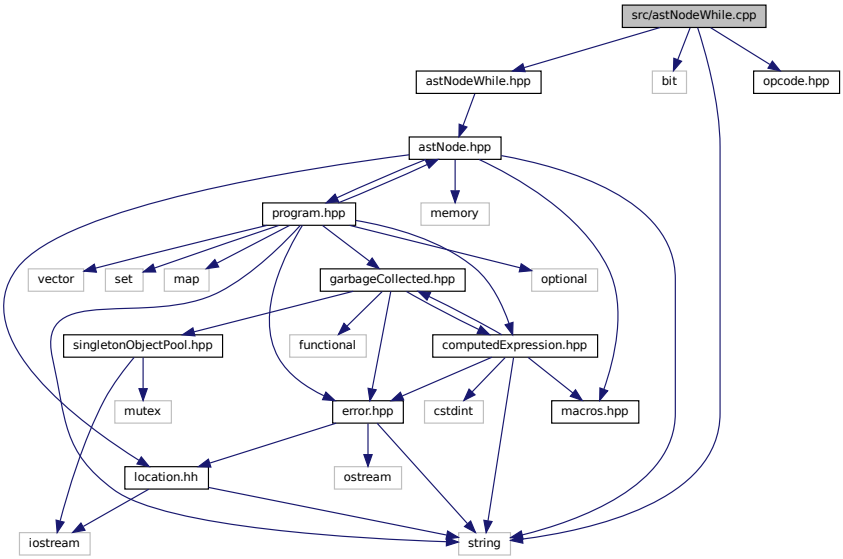
6.66 src/astNodeWhile.cpp File Reference

Define the Tang::AstNodeWhile class.

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

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

Include dependency graph for astNodeWhile.cpp:



6.66.1 Detailed Description

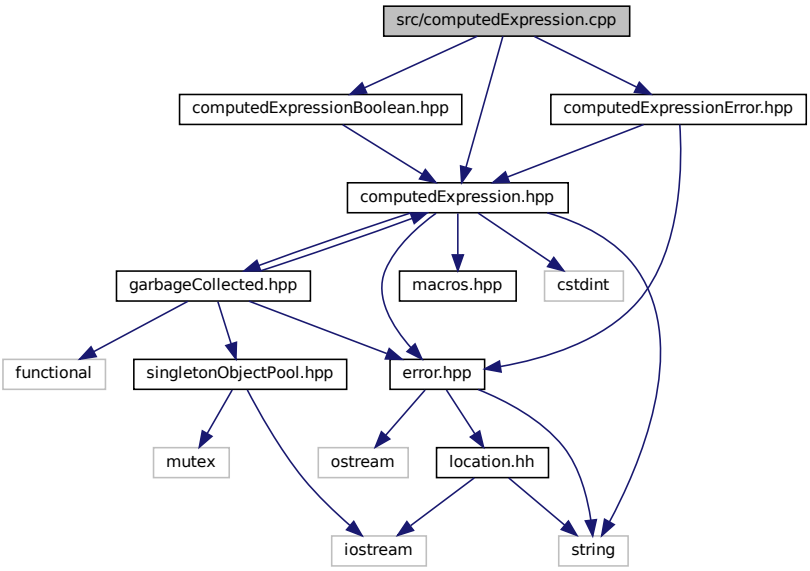
Define the Tang::AstNodeWhile class.

6.67 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:



6.67.1 Detailed Description

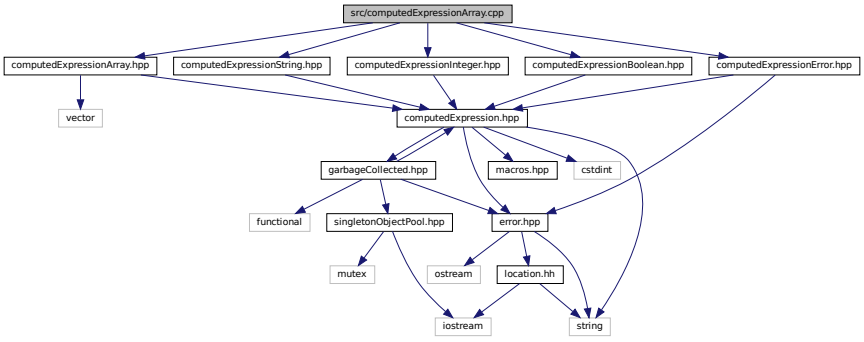
Define the Tang::ComputedExpression class.

6.68 src/computedExpressionArray.cpp File Reference

Define the Tang::ComputedExpressionArray class.

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

Include dependency graph for computedExpressionArray.cpp:



6.68.1 Detailed Description

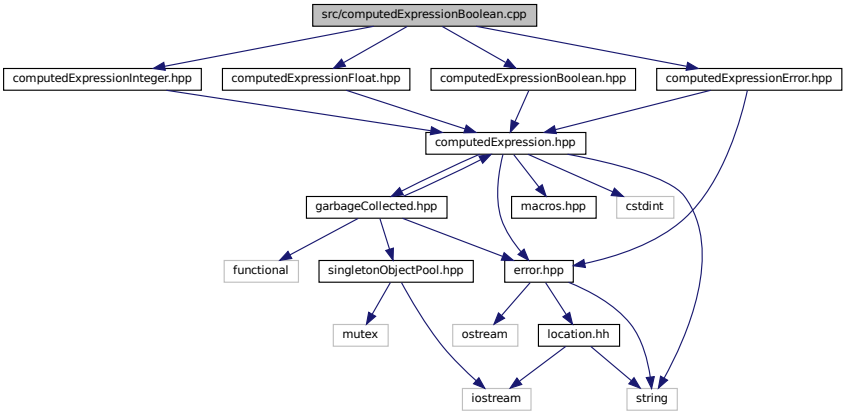
Define the Tang::ComputedExpressionArray class.

6.69 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:



6.69.1 Detailed Description

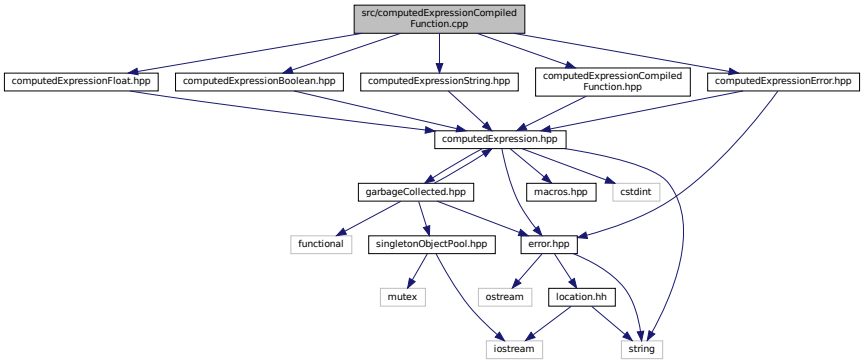
Define the Tang::ComputedExpressionBoolean class.

6.70 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:



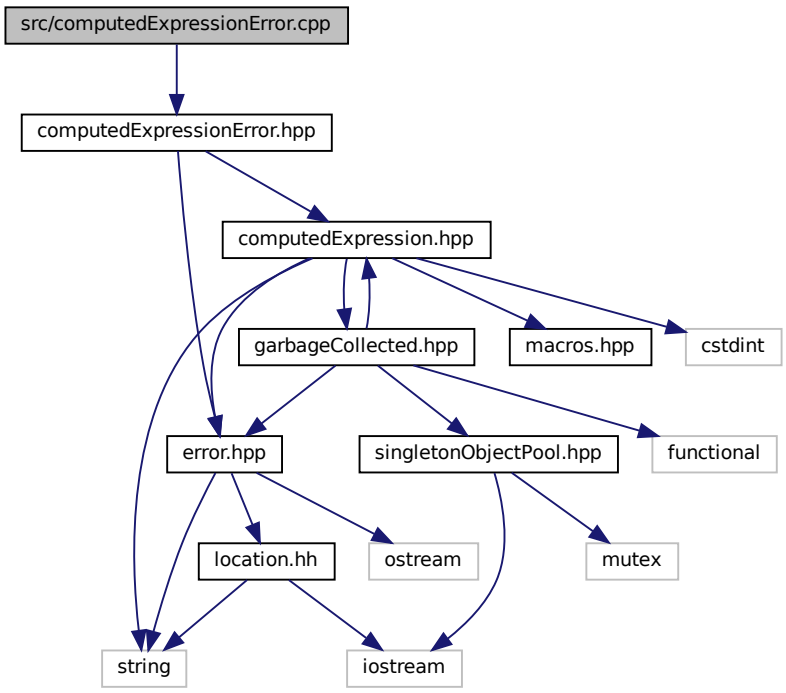
6.70.1 Detailed Description

Define the Tang::ComputedExpressionCompiledFunction class.

6.71 src/computedExpressionError.cpp File Reference

Define the Tang::ComputedExpressionError class.

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



6.71.1 Detailed Description

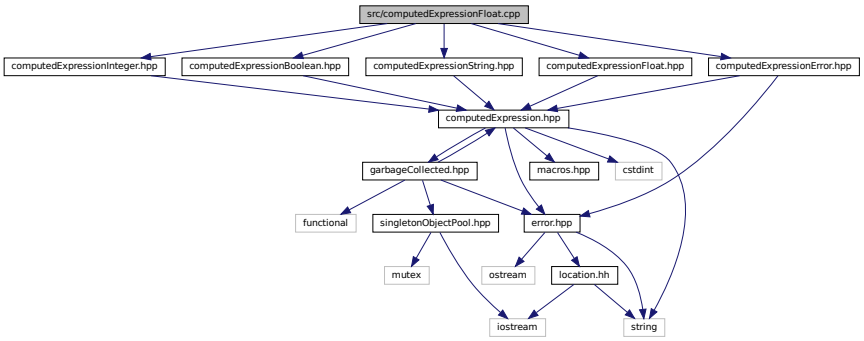
Define the Tang::ComputedExpressionError class.

6.72 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:



6.72.1 Detailed Description

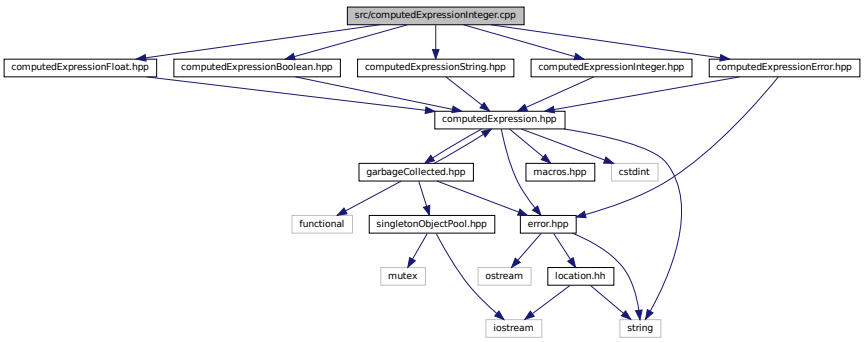
Define the Tang::ComputedExpressionFloat class.

6.73 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:



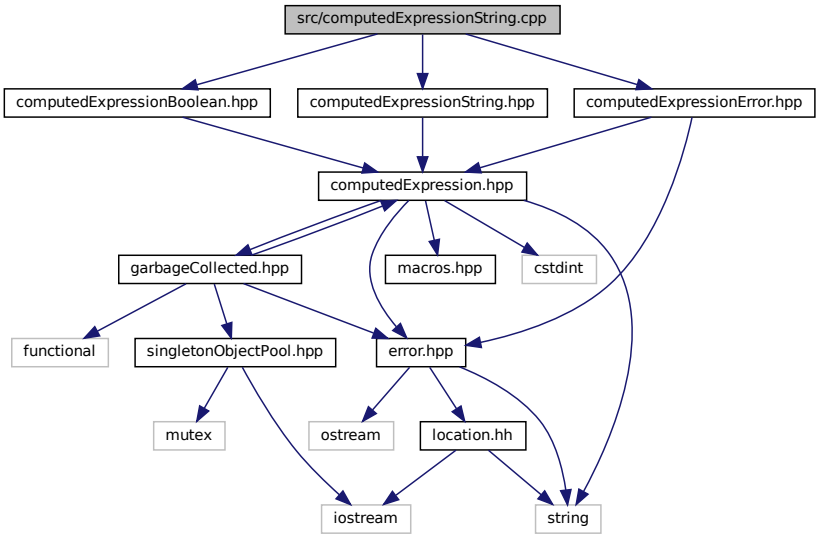
6.73.1 Detailed Description

Define the Tang::ComputedExpressionInteger class.

6.74 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:



6.74.1 Detailed Description

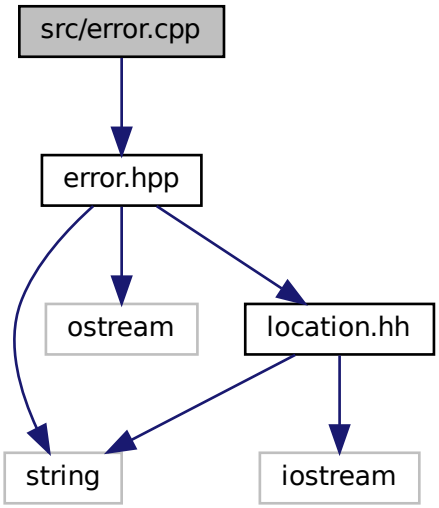
Define the Tang::ComputedExpressionString class.

6.75 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)

6.75.1 Detailed Description

Define the Tang::Error class.

6.75.2 Function Documentation

6.75.2.1 operator<<()

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

Parameters

out	The output stream.
error	The Error object.

Returns

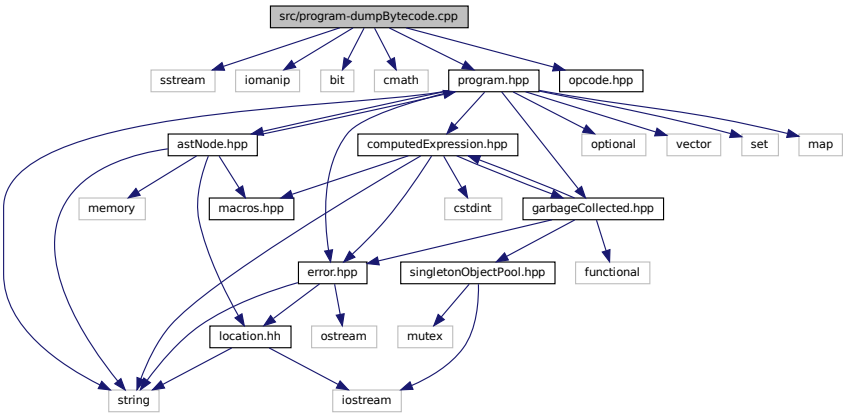
The output stream.

6.76 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.

6.76.1 Detailed Description

Define the Tang::Program::dumpBytecode method.

6.76.2 Macro Definition Documentation

6.76.2.1 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

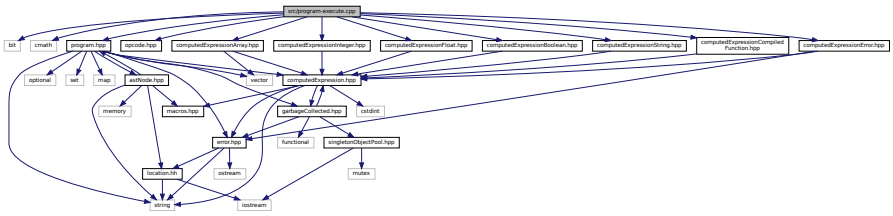
x	The number of additional vector entries that should exist.

6.77 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 "computedExpressionArray.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.

6.77.1 Detailed Description

Define the Tang::Program::execute method.

6.77.2 Macro Definition Documentation

6.77.2.1 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.

6.77.2.2 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

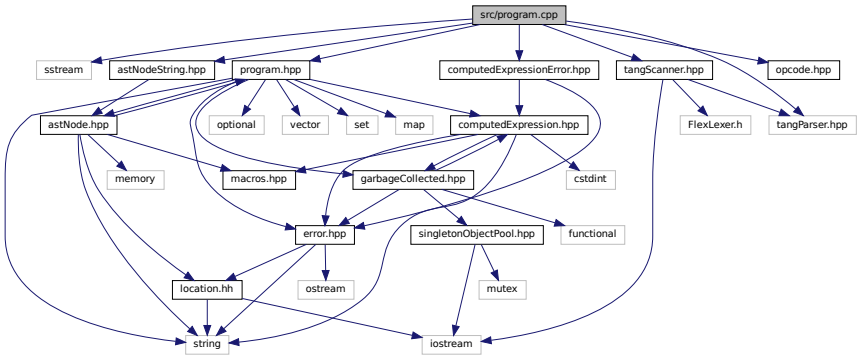
x	The number of entries that should exist in the stack.

6.78 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:



6.78.1 Detailed Description

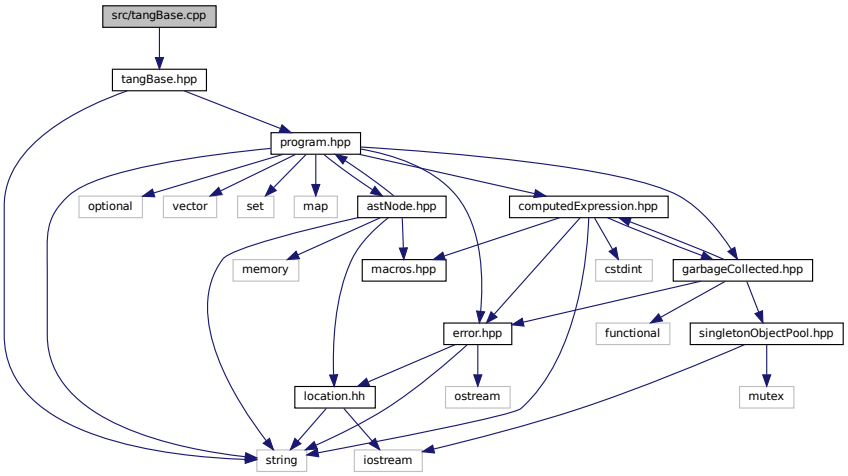
Define the Tang::Program class.

6.79 src/tangBase.cpp File Reference

Define the Tang::TangBase class.

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

Include dependency graph for tangBase.cpp:



6.79.1 Detailed Description

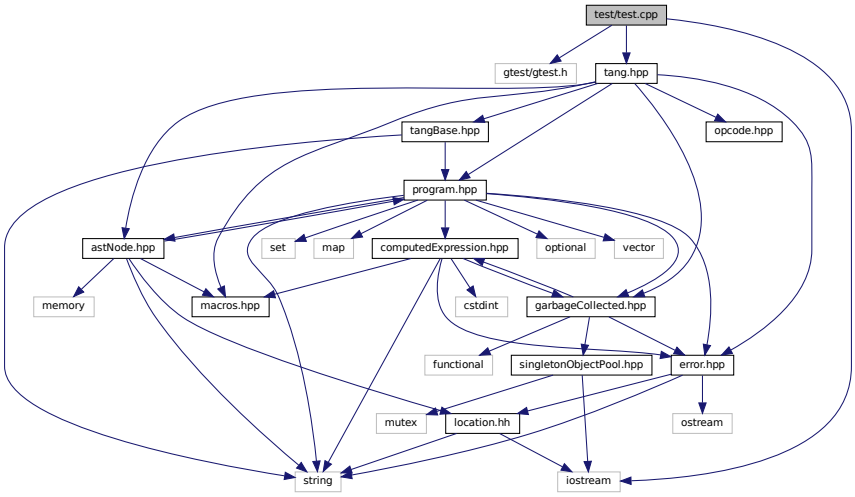
Define the Tang::TangBase class.

6.80 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** (Expression, ArrayIndex)
- **TEST** (CodeBlock, Statements)
- **TEST** (Assign, Identifier)
- **TEST** (Assign, Index)
- **TEST** (ControlFlow, IfElse)
- **TEST** (ControlFlow, While)
- **TEST** (ControlFlow, Break)
- **TEST** (ControlFlow, Continue)
- **TEST** (ControlFlow, DoWhile)
- **TEST** (ControlFlow, For)
- **TEST** (Print, Default)
- **TEST** (Function, Compiled)
- **TEST** (Function, Recursion)
- **TEST** (Function, FunctionCall)
- **TEST** (Function, Return)
- **TEST** (Function, PassByValueVsRef)
- int **main** (int argc, char **argv)

6.80.1 Detailed Description

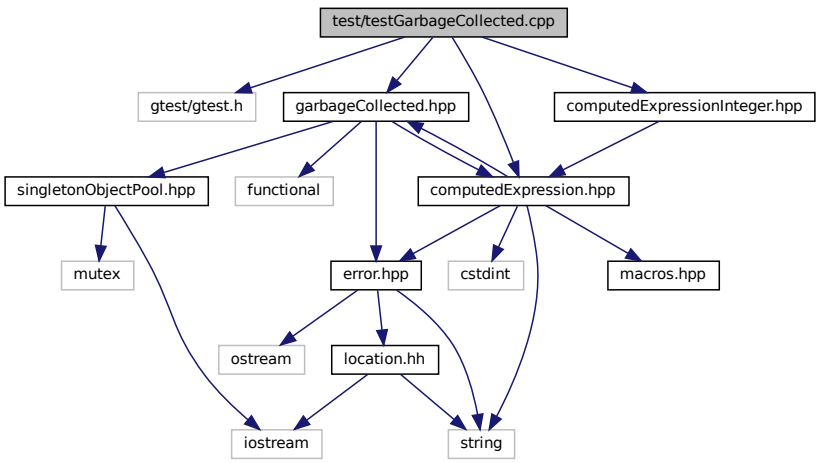
Test the general language behaviors.

6.81 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)

6.81.1 Detailed Description

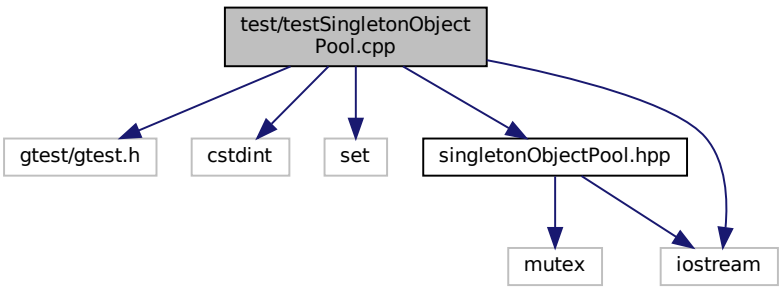
Test the generic behavior of the Tang::GarbageCollected class.

6.82 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)

6.82.1 Detailed Description

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

Index

__add
Tang::ComputedExpression, 101
Tang::ComputedExpressionArray, 112
Tang::ComputedExpressionBoolean, 123
Tang::ComputedExpressionCompiledFunction, 134
Tang::ComputedExpressionError, 145
Tang::ComputedExpressionFloat, 156
Tang::ComputedExpressionInteger, 167
Tang::ComputedExpressionString, 178
__assign_index
Tang::ComputedExpression, 102
Tang::ComputedExpressionArray, 112
Tang::ComputedExpressionBoolean, 123
Tang::ComputedExpressionCompiledFunction, 134
Tang::ComputedExpressionError, 145
Tang::ComputedExpressionFloat, 156
Tang::ComputedExpressionInteger, 167
Tang::ComputedExpressionString, 178
__boolean
Tang::ComputedExpression, 102
Tang::ComputedExpressionArray, 113
Tang::ComputedExpressionBoolean, 124
Tang::ComputedExpressionCompiledFunction, 135
Tang::ComputedExpressionError, 146
Tang::ComputedExpressionFloat, 157
Tang::ComputedExpressionInteger, 168
Tang::ComputedExpressionString, 180
__divide
Tang::ComputedExpression, 102
Tang::ComputedExpressionArray, 113
Tang::ComputedExpressionBoolean, 124
Tang::ComputedExpressionCompiledFunction, 135
Tang::ComputedExpressionError, 146
Tang::ComputedExpressionFloat, 157
Tang::ComputedExpressionInteger, 168
Tang::ComputedExpressionString, 180
__equal
Tang::ComputedExpression, 103
Tang::ComputedExpressionArray, 114
Tang::ComputedExpressionBoolean, 125
Tang::ComputedExpressionCompiledFunction, 135
Tang::ComputedExpressionError, 146
Tang::ComputedExpressionFloat, 158
Tang::ComputedExpressionInteger, 169
Tang::ComputedExpressionString, 180
__float
Tang::ComputedExpression, 103
Tang::ComputedExpressionArray, 114
Tang::ComputedExpressionBoolean, 125
Tang::ComputedExpressionCompiledFunction, 136
Tang::ComputedExpressionError, 147
Tang::ComputedExpressionFloat, 158
Tang::ComputedExpressionInteger, 169
Tang::ComputedExpressionString, 181
__index
Tang::ComputedExpression, 103
Tang::ComputedExpressionArray, 114
Tang::ComputedExpressionBoolean, 125
Tang::ComputedExpressionCompiledFunction, 136
Tang::ComputedExpressionError, 147
Tang::ComputedExpressionFloat, 158
Tang::ComputedExpressionInteger, 169
Tang::ComputedExpressionString, 181
__integer
Tang::ComputedExpression, 104
Tang::ComputedExpressionArray, 115
Tang::ComputedExpressionBoolean, 126
Tang::ComputedExpressionCompiledFunction, 137
Tang::ComputedExpressionError, 147
Tang::ComputedExpressionFloat, 159
Tang::ComputedExpressionInteger, 170
Tang::ComputedExpressionString, 182
__lessThan
Tang::ComputedExpression, 104
Tang::ComputedExpressionArray, 115
Tang::ComputedExpressionBoolean, 126
Tang::ComputedExpressionCompiledFunction, 137
Tang::ComputedExpressionError, 148
Tang::ComputedExpressionFloat, 159
Tang::ComputedExpressionInteger, 170
Tang::ComputedExpressionString, 182
__modulo
Tang::ComputedExpression, 104
Tang::ComputedExpressionArray, 115
Tang::ComputedExpressionBoolean, 126
Tang::ComputedExpressionCompiledFunction, 137
Tang::ComputedExpressionError, 148
Tang::ComputedExpressionFloat, 159
Tang::ComputedExpressionInteger, 170
Tang::ComputedExpressionString, 182
__multiply
Tang::ComputedExpression, 105
Tang::ComputedExpressionArray, 116
Tang::ComputedExpressionBoolean, 127
Tang::ComputedExpressionCompiledFunction, 138
Tang::ComputedExpressionError, 148
Tang::ComputedExpressionFloat, 160
Tang::ComputedExpressionInteger, 171

Tang::ComputedExpressionString, 183
__negative
Tang::ComputedExpression, 105
Tang::ComputedExpressionArray, 116
Tang::ComputedExpressionBoolean, 127
Tang::ComputedExpressionCompiledFunction, 138
Tang::ComputedExpressionError, 149
Tang::ComputedExpressionFloat, 160
Tang::ComputedExpressionInteger, 171
Tang::ComputedExpressionString, 183
__not
Tang::ComputedExpression, 105
Tang::ComputedExpressionArray, 116
Tang::ComputedExpressionBoolean, 127
Tang::ComputedExpressionCompiledFunction, 138
Tang::ComputedExpressionError, 149
Tang::ComputedExpressionFloat, 160
Tang::ComputedExpressionInteger, 171
Tang::ComputedExpressionString, 183
__string
Tang::ComputedExpression, 106
Tang::ComputedExpressionArray, 117
Tang::ComputedExpressionBoolean, 128
Tang::ComputedExpressionCompiledFunction, 138
Tang::ComputedExpressionError, 149
Tang::ComputedExpressionFloat, 161
Tang::ComputedExpressionInteger, 172
Tang::ComputedExpressionString, 183
__subtract
Tang::ComputedExpression, 106
Tang::ComputedExpressionArray, 117
Tang::ComputedExpressionBoolean, 128
Tang::ComputedExpressionCompiledFunction, 139
Tang::ComputedExpressionError, 149
Tang::ComputedExpressionFloat, 161
Tang::ComputedExpressionInteger, 172
Tang::ComputedExpressionString, 184
∼GarbageCollected
Tang::GarbageCollected, 193

ADD
opcode.hpp, 262
Add
Tang::AstNodeBinary, 24
addBreak
Tang::Program, 212
addBytecode
Tang::Program, 212
addContinue
Tang::Program, 212
addIdentifier
Tang::Program, 212
addIdentifierAssigned
Tang::Program, 213
addString
Tang::Program, 213
And
Tang::AstNodeBinary, 24
ARRAY
opcode.hpp, 262
ASSIGNINDEX
opcode.hpp, 262
AstNode
Tang::AstNode, 13
AstNodeArray
Tang::AstNodeArray, 17
AstNodeAssign
Tang::AstNodeAssign, 20
AstNodeBinary
Tang::AstNodeBinary, 24
AstNodeBlock
Tang::AstNodeBlock, 28
AstNodeBoolean
Tang::AstNodeBoolean, 31
AstNodeBreak
Tang::AstNodeBreak, 35
AstNodeCast
Tang::AstNodeCast, 39
AstNodeContinue
Tang::AstNodeContinue, 42
AstNodeDoWhile
Tang::AstNodeDoWhile, 46
AstNodeFloat
Tang::AstNodeFloat, 49
AstNodeFor
Tang::AstNodeFor, 53
AstNodeFunctionCall
Tang::AstNodeFunctionCall, 56
AstNodeFunctionDeclaration
Tang::AstNodeFunctionDeclaration, 60
AstNodeIdentifier
Tang::AstNodeIdentifier, 64
AstNodeIfElse
Tang::AstNodeIfElse, 68
AstNodeIndex
Tang::AstNodeIndex, 71
AstNodeInteger
Tang::AstNodeInteger, 75
AstNodePrint
Tang::AstNodePrint, 79
AstNodeReturn
Tang::AstNodeReturn, 82
AstNodeString
Tang::AstNodeString, 86
AstNodeTernary
Tang::AstNodeTernary, 90
AstNodeUnary
Tang::AstNodeUnary, 94
AstNodeWhile
Tang::AstNodeWhile, 97

BOOLEAN
opcode.hpp, 262
Boolean
Tang::AstNodeCast, 39
build/generated/location.hh, 225

CALLFUNC

opcode.hpp, 262
CASTBOOLEAN
opcode.hpp, 262
CASTFLOAT
opcode.hpp, 262
CASTINTEGER
opcode.hpp, 262
CodeType
Tang::Program, 211
compile
Tang::AstNode, 14
Tang::AstNodeArray, 17
Tang::AstNodeAssign, 21
Tang::AstNodeBinary, 25
Tang::AstNodeBlock, 28
Tang::AstNodeBoolean, 31
Tang::AstNodeBreak, 36
Tang::AstNodeCast, 39
Tang::AstNodeContinue, 43
Tang::AstNodeDoWhile, 46
Tang::AstNodeFloat, 50
Tang::AstNodeFor, 53
Tang::AstNodeFunctionCall, 57
Tang::AstNodeFunctionDeclaration, 60
Tang::AstNodeIdentifier, 64
Tang::AstNodeIfElse, 68
Tang::AstNodeIndex, 72
Tang::AstNodeInteger, 76
Tang::AstNodePrint, 79
Tang::AstNodeReturn, 83
Tang::AstNodeString, 86
Tang::AstNodeTernary, 90
Tang::AstNodeUnary, 94
Tang::AstNodeWhile, 97
compileLiteral
Tang::AstNodeString, 87
compilePreprocess
Tang::AstNode, 14
Tang::AstNodeArray, 18
Tang::AstNodeAssign, 21
Tang::AstNodeBinary, 25
Tang::AstNodeBlock, 29
Tang::AstNodeBoolean, 33
Tang::AstNodeBreak, 36
Tang::AstNodeCast, 40
Tang::AstNodeContinue, 43
Tang::AstNodeDoWhile, 47
Tang::AstNodeFloat, 50
Tang::AstNodeFor, 54
Tang::AstNodeFunctionCall, 57
Tang::AstNodeFunctionDeclaration, 61
Tang::AstNodeIdentifier, 65
Tang::AstNodeIfElse, 69
Tang::AstNodeIndex, 72
Tang::AstNodeInteger, 76
Tang::AstNodePrint, 80
Tang::AstNodeReturn, 83
Tang::AstNodeString, 87
Tang::AstNodeTernary, 91
Tang::AstNodeUnary, 95
Tang::AstNodeWhile, 98
compileScript
Tang::TangBase, 220
ComputedExpressionArray
Tang::ComputedExpressionArray, 112
ComputedExpressionBoolean
Tang::ComputedExpressionBoolean, 123
ComputedExpressionCompiledFunction
Tang::ComputedExpressionCompiledFunction, 134
ComputedExpressionError
Tang::ComputedExpressionError, 144
ComputedExpressionFloat
Tang::ComputedExpressionFloat, 156
ComputedExpressionInteger
Tang::ComputedExpressionInteger, 167
ComputedExpressionString
Tang::ComputedExpressionString, 178
COPY
opcode.hpp, 262

Default
Tang::AstNode, 13
Tang::AstNodeArray, 17
Tang::AstNodeAssign, 20
Tang::AstNodeBinary, 24
Tang::AstNodeBlock, 28
Tang::AstNodeBoolean, 31
Tang::AstNodeBreak, 35
Tang::AstNodeCast, 39
Tang::AstNodeContinue, 42
Tang::AstNodeDoWhile, 46
Tang::AstNodeFloat, 49
Tang::AstNodeFor, 53
Tang::AstNodeFunctionCall, 56
Tang::AstNodeFunctionDeclaration, 60
Tang::AstNodeIdentifier, 64
Tang::AstNodeIfElse, 68
Tang::AstNodeIndex, 71
Tang::AstNodeInteger, 75
Tang::AstNodePrint, 79
Tang::AstNodeReturn, 82
Tang::AstNodeString, 86
Tang::AstNodeTernary, 90
Tang::AstNodeUnary, 94
Tang::AstNodeWhile, 97
DIVIDE
opcode.hpp, 262
Divide
Tang::AstNodeBinary, 24
dump
Tang::AstNode, 15
Tang::AstNodeArray, 18
Tang::AstNodeAssign, 22
Tang::AstNodeBinary, 26
Tang::AstNodeBlock, 29
Tang::AstNodeBoolean, 33
Tang::AstNodeBreak, 37

Tang::AstNodeCast, 40
Tang::AstNodeContinue, 44
Tang::AstNodeDoWhile, 47
Tang::AstNodeFloat, 51
Tang::AstNodeFor, 54
Tang::AstNodeFunctionCall, 58
Tang::AstNodeFunctionDeclaration, 61
Tang::AstNodeIdentifier, 65
Tang::AstNodeIfElse, 69
Tang::AstNodeIndex, 73
Tang::AstNodeInteger, 77
Tang::AstNodePrint, 80
Tang::AstNodeReturn, 84
Tang::AstNodeString, 88
Tang::AstNodeTernary, 91
Tang::AstNodeUnary, 95
Tang::AstNodeWhile, 98
Tang::ComputedExpression, 106
Tang::ComputedExpressionArray, 117
Tang::ComputedExpressionBoolean, 128
Tang::ComputedExpressionCompiledFunction, 139
Tang::ComputedExpressionError, 150
Tang::ComputedExpressionFloat, 162
Tang::ComputedExpressionInteger, 173
Tang::ComputedExpressionString, 184
dumpBytecode
Tang::Program, 213
DUMPPROGRAMCHECK
program-dumpBytecode.cpp, 292

EQ
opcode.hpp, 262
Equal
Tang::AstNodeBinary, 24
Error
Tang::Error, 189
error.cpp
operator<<, 291
execute
Tang::Program, 214
EXECUTEPROGRAMCHECK
program-execute.cpp, 294

FLOAT
opcode.hpp, 262
Float
Tang::AstNodeCast, 39
FUNCTION
opcode.hpp, 262
functionsDeclared
Tang::Program, 218

GarbageCollected
Tang::GarbageCollected, 192, 193
get
Tang::SingletonObjectPool< T >, 219
get_next_token
Tang::TangScanner, 222
getAst
Tang::Program, 214
getBytecode
Tang::Program, 214
getCode
Tang::Program, 214
getCollection
Tang::AstNodeIndex, 73
getIdentifiers
Tang::Program, 215
getIdentifiersAssigned
Tang::Program, 215
getIndex
Tang::AstNodeIndex, 73
getInstance
Tang::SingletonObjectPool< T >, 219
getResult
Tang::Program, 215
getStrings
Tang::Program, 215
GreaterThan
Tang::AstNodeBinary, 24
GreaterThanEqual
Tang::AstNodeBinary, 24
GT
opcode.hpp, 262
GTE
opcode.hpp, 262

include/astNode.hpp, 227
include/astNodeArray.hpp, 228
include/astNodeAssign.hpp, 229
include/astNodeBinary.hpp, 230
include/astNodeBlock.hpp, 231
include/astNodeBoolean.hpp, 232
include/astNodeBreak.hpp, 233
include/astNodeCast.hpp, 234
include/astNodeContinue.hpp, 235
include/astNodeDoWhile.hpp, 236
include/astNodeFloat.hpp, 237
include/astNodeFor.hpp, 238
include/astNodeFunctionCall.hpp, 239
include/astNodeFunctionDeclaration.hpp, 240
include/astNodeIdentifier.hpp, 241
include/astNodeIfElse.hpp, 242
include/astNodeIndex.hpp, 243
include/astNodeInteger.hpp, 244
include/astNodePrint.hpp, 245
include/astNodeReturn.hpp, 246
include/astNodeString.hpp, 247
include/astNodeTernary.hpp, 248
include/astNodeUnary.hpp, 249
include/astNodeWhile.hpp, 250
include/computedExpression.hpp, 251
include/computedExpressionArray.hpp, 252
include/computedExpressionBoolean.hpp, 253
include/computedExpressionCompiledFunction.hpp, 254
include/computedExpressionError.hpp, 255
include/computedExpressionFloat.hpp, 256

include/computedExpressionInteger.hpp, 257
include/computedExpressionString.hpp, 258
include/error.hpp, 259
include/garbageCollected.hpp, 260
include/macros.hpp, 260
include/opcode.hpp, 261
include/program.hpp, 262
include/singletonObjectPool.hpp, 264
include/tang.hpp, 265
include/tangBase.hpp, 266
include/tangScanner.hpp, 267
INDEX
    opcode.hpp, 262
INTEGER
    opcode.hpp, 262
Integer
    Tang::AstNodeCast, 39
is_equal
    Tang::ComputedExpression, 107–109
    Tang::ComputedExpressionArray, 118–120
    Tang::ComputedExpressionBoolean, 129–131
    Tang::ComputedExpressionCompiledFunction, 139–141
    Tang::ComputedExpressionError, 150, 152, 153
    Tang::ComputedExpressionFloat, 162–164
    Tang::ComputedExpressionInteger, 173–175
    Tang::ComputedExpressionString, 184–186
IsAssignment
    Tang::AstNode, 13
    Tang::AstNodeArray, 17
    Tang::AstNodeAssign, 20
    Tang::AstNodeBinary, 24
    Tang::AstNodeBlock, 28
    Tang::AstNodeBoolean, 31
    Tang::AstNodeBreak, 35
    Tang::AstNodeCast, 39
    Tang::AstNodeContinue, 42
    Tang::AstNodeDoWhile, 46
    Tang::AstNodeFloat, 49
    Tang::AstNodeFor, 53
    Tang::AstNodeFunctionCall, 56
    Tang::AstNodeFunctionDeclaration, 60
    Tang::AstNodeIdentifier, 64
    Tang::AstNodeIfElse, 68
    Tang::AstNodeIndex, 71
    Tang::AstNodeInteger, 75
    Tang::AstNodePrint, 79
    Tang::AstNodeReturn, 82
    Tang::AstNodeString, 86
    Tang::AstNodeTernary, 90
    Tang::AstNodeUnary, 94
    Tang::AstNodeWhile, 97
isCopyNeeded
    Tang::ComputedExpression, 109
    Tang::ComputedExpressionArray, 120
    Tang::ComputedExpressionBoolean, 131
    Tang::ComputedExpressionCompiledFunction, 142
    Tang::ComputedExpressionError, 153
    Tang::ComputedExpressionFloat, 164
    Tang::ComputedExpressionInteger, 175
    Tang::ComputedExpressionString, 187
    Tang::GarbageCollected, 193

JMP
    opcode.hpp, 262
JMPF
    opcode.hpp, 262
JMPF_POP
    opcode.hpp, 262
JMPT
    opcode.hpp, 262
JMPT_POP
    opcode.hpp, 262

LessThan
    Tang::AstNodeBinary, 24
LessThanEqual
    Tang::AstNodeBinary, 24
location.hh
    operator<<, 226, 227
LT
    opcode.hpp, 262
LTE
    opcode.hpp, 262

make
    Tang::GarbageCollected, 194
makeCopy
    Tang::ComputedExpression, 109
    Tang::ComputedExpressionArray, 120
    Tang::ComputedExpressionBoolean, 131
    Tang::ComputedExpressionCompiledFunction, 142
    Tang::ComputedExpressionError, 153
    Tang::ComputedExpressionFloat, 164
    Tang::ComputedExpressionInteger, 175
    Tang::ComputedExpressionString, 187
    Tang::GarbageCollected, 194
MODULO
    opcode.hpp, 262
Modulo
    Tang::AstNodeBinary, 24
MULTIPLY
    opcode.hpp, 262
Multiply
    Tang::AstNodeBinary, 24

NEGATIVE
    opcode.hpp, 262
Negative
    Tang::AstNodeUnary, 93
NEQ
    opcode.hpp, 262
NOT
    opcode.hpp, 262
Not
    Tang::AstNodeUnary, 93
NotEqual

Tang::AstNodeBinary, 24
NULLVAL
opcode.hpp, 262

Opcode
opcode.hpp, 261
opcode.hpp
ADD, 262
ARRAY, 262
ASSIGNINDEX, 262
BOOLEAN, 262
CALLFUNC, 262
CASTBOOLEAN, 262
CASTFLOAT, 262
CASTINTEGER, 262
COPY, 262
DIVIDE, 262
EQ, 262
FLOAT, 262
FUNCTION, 262
GT, 262
GTE, 262
INDEX, 262
INTEGER, 262
JMP, 262
JMPF, 262
JMPF_POP, 262
JMPT, 262
JMPT_POP, 262
LT, 262
LTE, 262
MODULO, 262
MULTIPLY, 262
NEGATIVE, 262
NEQ, 262
NOT, 262
NULLVAL, 262
Opcode, 261
PEEK, 262
POKE, 262
POP, 262
PRINT, 262
RETURN, 262
STRING, 262
SUBTRACT, 262
Operation
Tang::AstNodeBinary, 23
Operator
Tang::AstNodeUnary, 93
operator!
Tang::GarbageCollected, 195
operator!=
Tang::GarbageCollected, 195
operator<
Tang::GarbageCollected, 200
operator<<
error.cpp, 291
location.hh, 226, 227
Tang::Error, 189
Tang::GarbageCollected, 205
operator<=
Tang::GarbageCollected, 200
operator>
Tang::GarbageCollected, 204
operator>=
Tang::GarbageCollected, 205
operator*
Tang::GarbageCollected, 196, 197
operator+
Tang::GarbageCollected, 197
operator-
Tang::GarbageCollected, 198
operator->
Tang::GarbageCollected, 199
operator/
Tang::GarbageCollected, 199
operator=
Tang::GarbageCollected, 201
operator==
Tang::GarbageCollected, 201–204
operator%
Tang::GarbageCollected, 196
Or
Tang::AstNodeBinary, 24

PEEK
opcode.hpp, 262
POKE
opcode.hpp, 262
POP
opcode.hpp, 262
popBreakStack
Tang::Program, 216
popContinueStack
Tang::Program, 216
PreprocessState
Tang::AstNode, 13
Tang::AstNodeArray, 17
Tang::AstNodeAssign, 20
Tang::AstNodeBinary, 24
Tang::AstNodeBlock, 27
Tang::AstNodeBoolean, 31
Tang::AstNodeBreak, 35
Tang::AstNodeCast, 38
Tang::AstNodeContinue, 42
Tang::AstNodeDoWhile, 45
Tang::AstNodeFloat, 49
Tang::AstNodeFor, 52
Tang::AstNodeFunctionCall, 56
Tang::AstNodeFunctionDeclaration, 59
Tang::AstNodeIdentifier, 64
Tang::AstNodeIfElse, 67
Tang::AstNodeIndex, 71
Tang::AstNodeInteger, 75
Tang::AstNodePrint, 78
Tang::AstNodeReturn, 82
Tang::AstNodeString, 85
Tang::AstNodeTernary, 90

Tang::AstNodeUnary, 93
Tang::AstNodeWhile, 97
PRINT
opcode.hpp, 262
Program
Tang::Program, 211
program-dumpBytecode.cpp
DUMPPROGRAMCHECK, 292
program-execute.cpp
EXECUTEPROGRAMCHECK, 294
STACKCHECK, 294
pushEnvironment
Tang::Program, 217

recycle
Tang::SingletonObjectPool< T >, 219
RETURN
opcode.hpp, 262

Script
Tang::Program, 211
setFunctionStackDeclaration
Tang::Program, 217
setJumpTarget
Tang::Program, 218
src/astNode.cpp, 268
src/astNodeArray.cpp, 268
src/astNodeAssign.cpp, 269
src/astNodeBinary.cpp, 270
src/astNodeBlock.cpp, 271
src/astNodeBoolean.cpp, 271
src/astNodeBreak.cpp, 272
src/astNodeCast.cpp, 273
src/astNodeContinue.cpp, 273
src/astNodeDoWhile.cpp, 274
src/astNodeFloat.cpp, 275
src/astNodeFor.cpp, 276
src/astNodeFunctionCall.cpp, 276
src/astNodeFunctionDeclaration.cpp, 277
src/astNodeIdentifier.cpp, 278
src/astNodeIfElse.cpp, 279
src/astNodeIndex.cpp, 279
src/astNodeInteger.cpp, 280
src/astNodePrint.cpp, 281
src/astNodeReturn.cpp, 281
src/astNodeString.cpp, 282
src/astNodeTernary.cpp, 283
src/astNodeUnary.cpp, 284
src/astNodeWhile.cpp, 284
src/computedExpression.cpp, 285
src/computedExpressionArray.cpp, 286
src/computedExpressionBoolean.cpp, 287
src/computedExpressionCompiledFunction.cpp, 287
src/computedExpressionError.cpp, 288
src/computedExpressionFloat.cpp, 289
src/computedExpressionInteger.cpp, 289
src/computedExpressionString.cpp, 290
src/error.cpp, 291
src/program-dumpBytecode.cpp, 292
src/program-execute.cpp, 293
src/program.cpp, 294
src/tangBase.cpp, 295
STACKCHECK
program-execute.cpp, 294
STRING
opcode.hpp, 262
SUBTRACT
opcode.hpp, 262
Subtract
Tang::AstNodeBinary, 24

Tang::AstNode, 11
AstNode, 13
compile, 14
compilePreprocess, 14
Default, 13
dump, 15
IsAssignment, 13
PreprocessState, 13
Tang::AstNodeArray, 15
AstNodeArray, 17
compile, 17
compilePreprocess, 18
Default, 17
dump, 18
IsAssignment, 17
PreprocessState, 17
Tang::AstNodeAssign, 19
AstNodeAssign, 20
compile, 21
compilePreprocess, 21
Default, 20
dump, 22
IsAssignment, 20
PreprocessState, 20
Tang::AstNodeBinary, 22
Add, 24
And, 24
AstNodeBinary, 24
compile, 25
compilePreprocess, 25
Default, 24
Divide, 24
dump, 26
Equal, 24
GreaterThan, 24
GreaterThanEqual, 24
IsAssignment, 24
LessThan, 24
LessThanEqual, 24
Modulo, 24
Multiply, 24
NotEqual, 24
Operation, 23
Or, 24
PreprocessState, 24
Subtract, 24
Tang::AstNodeBlock, 26

AstNodeBlock, 28
compile, 28
compilePreprocess, 29
Default, 28
dump, 29
IsAssignment, 28
PreprocessState, 27
Tang::AstNodeBoolean, 30
AstNodeBoolean, 31
compile, 31
compilePreprocess, 33
Default, 31
dump, 33
IsAssignment, 31
PreprocessState, 31
Tang::AstNodeBreak, 34
AstNodeBreak, 35
compile, 36
compilePreprocess, 36
Default, 35
dump, 37
IsAssignment, 35
PreprocessState, 35
Tang::AstNodeCast, 37
AstNodeCast, 39
Boolean, 39
compile, 39
compilePreprocess, 40
Default, 39
dump, 40
Float, 39
Integer, 39
IsAssignment, 39
PreprocessState, 38
Type, 39
Tang::AstNodeContinue, 41
AstNodeContinue, 42
compile, 43
compilePreprocess, 43
Default, 42
dump, 44
IsAssignment, 42
PreprocessState, 42
Tang::AstNodeDoWhile, 44
AstNodeDoWhile, 46
compile, 46
compilePreprocess, 47
Default, 46
dump, 47
IsAssignment, 46
PreprocessState, 45
Tang::AstNodeFloat, 48
AstNodeFloat, 49
compile, 50
compilePreprocess, 50
Default, 49
dump, 51
IsAssignment, 49
PreprocessState, 49
Tang::AstNodeFor, 51
AstNodeFor, 53
compile, 53
compilePreprocess, 54
Default, 53
dump, 54
IsAssignment, 53
PreprocessState, 52
Tang::AstNodeFunctionCall, 55
AstNodeFunctionCall, 56
compile, 57
compilePreprocess, 57
Default, 56
dump, 58
IsAssignment, 56
PreprocessState, 56
Tang::AstNodeFunctionDeclaration, 58
AstNodeFunctionDeclaration, 60
compile, 60
compilePreprocess, 61
Default, 60
dump, 61
IsAssignment, 60
PreprocessState, 59
Tang::AstNodeIdentifier, 62
AstNodeIdentifier, 64
compile, 64
compilePreprocess, 65
Default, 64
dump, 65
IsAssignment, 64
PreprocessState, 64
Tang::AstNodeIfElse, 66
AstNodeIfElse, 68
compile, 68
compilePreprocess, 69
Default, 68
dump, 69
IsAssignment, 68
PreprocessState, 67
Tang::AstNodeIndex, 70
AstNodeIndex, 71
compile, 72
compilePreprocess, 72
Default, 71
dump, 73
getCollection, 73
getIndex, 73
IsAssignment, 71
PreprocessState, 71
Tang::AstNodeInteger, 74
AstNodeInteger, 75
compile, 76
compilePreprocess, 76
Default, 75
dump, 77
IsAssignment, 75

PreprocessState, 75
Tang::AstNodePrint, 77
AstNodePrint, 79
compile, 79
compilePreprocess, 80
Default, 79
dump, 80
IsAssignment, 79
PreprocessState, 78
Type, 79
Tang::AstNodeReturn, 81
AstNodeReturn, 82
compile, 83
compilePreprocess, 83
Default, 82
dump, 84
IsAssignment, 82
PreprocessState, 82
Tang::AstNodeString, 84
AstNodeString, 86
compile, 86
compileLiteral, 87
compilePreprocess, 87
Default, 86
dump, 88
IsAssignment, 86
PreprocessState, 85
Tang::AstNodeTernary, 88
AstNodeTernary, 90
compile, 90
compilePreprocess, 91
Default, 90
dump, 91
IsAssignment, 90
PreprocessState, 90
Tang::AstNodeUnary, 92
AstNodeUnary, 94
compile, 94
compilePreprocess, 95
Default, 94
dump, 95
IsAssignment, 94
Negative, 93
Not, 93
Operator, 93
PreprocessState, 93
Tang::AstNodeWhile, 96
AstNodeWhile, 97
compile, 97
compilePreprocess, 98
Default, 97
dump, 98
IsAssignment, 97
PreprocessState, 97
Tang::ComputedExpression, 99
__add, 101
__assign_index, 102
__boolean, 102
__divide, 102
__equal, 103
__float, 103
__index, 103
__integer, 104
__lessThan, 104
__modulo, 104
__multiply, 105
__negative, 105
__not, 105
__string, 106
__subtract, 106
dump, 106
is_equal, 107–109
isCopyNeeded, 109
makeCopy, 109
Tang::ComputedExpressionArray, 110
__add, 112
__assign_index, 112
__boolean, 113
__divide, 113
__equal, 114
__float, 114
__index, 114
__integer, 115
__lessThan, 115
__modulo, 115
__multiply, 116
__negative, 116
__not, 116
__string, 117
__subtract, 117
ComputedExpressionArray, 112
dump, 117
is_equal, 118–120
isCopyNeeded, 120
makeCopy, 120
Tang::ComputedExpressionBoolean, 121
__add, 123
__assign_index, 123
__boolean, 124
__divide, 124
__equal, 125
__float, 125
__index, 125
__integer, 126
__lessThan, 126
__modulo, 126
__multiply, 127
__negative, 127
__not, 127
__string, 128
__subtract, 128
ComputedExpressionBoolean, 123
dump, 128
is_equal, 129–131
isCopyNeeded, 131
makeCopy, 131

Tang::ComputedExpressionCompiledFunction, 132
__add, 134
__assign_index, 134
__boolean, 135
__divide, 135
__equal, 135
__float, 136
__index, 136
__integer, 137
__lessThan, 137
__modulo, 137
__multiply, 138
__negative, 138
__not, 138
__string, 138
__subtract, 139
ComputedExpressionCompiledFunction, 134
dump, 139
is_equal, 139–141
isCopyNeeded, 142
makeCopy, 142
Tang::ComputedExpressionError, 143
__add, 145
__assign_index, 145
__boolean, 146
__divide, 146
__equal, 146
__float, 147
__index, 147
__integer, 147
__lessThan, 148
__modulo, 148
__multiply, 148
__negative, 149
__not, 149
__string, 149
__subtract, 149
ComputedExpressionError, 144
dump, 150
is_equal, 150, 152, 153
isCopyNeeded, 153
makeCopy, 153
Tang::ComputedExpressionFloat, 154
__add, 156
__assign_index, 156
__boolean, 157
__divide, 157
__equal, 158
__float, 158
__index, 158
__integer, 159
__lessThan, 159
__modulo, 159
__multiply, 160
__negative, 160
__not, 160
__string, 161
__subtract, 161
ComputedExpressionFloat, 156
dump, 162
is_equal, 162–164
isCopyNeeded, 164
makeCopy, 164
Tang::ComputedExpressionInteger, 165
__add, 167
__assign_index, 167
__boolean, 168
__divide, 168
__equal, 169
__float, 169
__index, 169
__integer, 170
__lessThan, 170
__modulo, 170
__multiply, 171
__negative, 171
__not, 171
__string, 172
__subtract, 172
ComputedExpressionInteger, 167
dump, 173
is_equal, 173–175
isCopyNeeded, 175
makeCopy, 175
Tang::ComputedExpressionString, 176
__add, 178
__assign_index, 178
__boolean, 180
__divide, 180
__equal, 180
__float, 181
__index, 181
__integer, 182
__lessThan, 182
__modulo, 182
__multiply, 183
__negative, 183
__not, 183
__string, 183
__subtract, 184
ComputedExpressionString, 178
dump, 184
is_equal, 184–186
isCopyNeeded, 187
makeCopy, 187
Tang::Error, 188
Error, 189
operator<<, 189
Tang::GarbageCollected, 190
~GarbageCollected, 193
GarbageCollected, 192, 193
isCopyNeeded, 193
make, 194
makeCopy, 194
operator!, 195
operator!=, 195

operator<, 200
operator<<, 205
operator<=, 200
operator>, 204
operator>=, 205
operator*, 196, 197
operator+, 197
operator-, 198
operator->, 199
operator/, 199
operator=, 201
operator==, 201–204
operator%, 196
Tang::location, 206
Tang::position, 208
Tang::Program, 209
addBreak, 212
addBytecode, 212
addContinue, 212
addIdentifier, 212
addIdentifierAssigned, 213
addString, 213
CodeType, 211
dumpBytecode, 213
execute, 214
functionsDeclared, 218
getAst, 214
getBytecode, 214
getCode, 214
getIdentifiers, 215
getIdentifiersAssigned, 215
getResult, 215
getStrings, 215
popBreakStack, 216
popContinueStack, 216
Program, 211
pushEnvironment, 217
Script, 211
setFunctionStackDeclaration, 217
setJumpTarget, 218
Template, 211
Tang::SingletonObjectPool< T >, 218
get, 219
getInstance, 219
recycle, 219
Tang::TangBase, 220
compileScript, 220
TangBase, 220
Tang::TangScanner, 221
get_next_token, 222
TangScanner, 222
TangBase
Tang::TangBase, 220
TangScanner
Tang::TangScanner, 222
Template
Tang::Program, 211
test/test.cpp, 296
test/testGarbageCollected.cpp, 297
test/testSingletonObjectPool.cpp, 298
Type
Tang::AstNodeCast, 39
Tang::AstNodePrint, 79