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 9
5.1 Tang::AstNode Class Reference . . . 9
5.1.1 Detailed Description . . . 11
5.1.2 Constructor & Destructor Documentation . . . 11
5.1.2.1 AstNode() . . . 11
5.1.3 Member Function Documentation . . . 11
5.1.3.1 makeCopy() . . . 11
5.2 Tang::AstNodeBinary Class Reference . . . 12
5.2.1 Detailed Description . . . 14
5.2.2 Constructor & Destructor Documentation . . . 14
5.2.2.1 AstNodeBinary() . . . 14
5.2.3 Member Function Documentation . . . 14
5.2.3.1 makeCopy() . . . 14
5.3 Tang::AstNodeBoolean Class Reference . . . 15
5.3.1 Detailed Description . . . 17
5.3.2 Constructor & Destructor Documentation . . . 17
5.3.2.1 AstNodeBoolean() . . . 17
5.3.3 Member Function Documentation . . . 17
5.3.3.1 makeCopy() . . . 17
5.4 Tang::AstNodeCastBoolean Class Reference . . . 18
5.4.1 Detailed Description . . . 20
5.4.2 Constructor & Destructor Documentation . . . 20
5.4.2.1 AstNodeCastBoolean() . . . 20
5.4.3 Member Function Documentation . . . 20
5.4.3.1 makeCopy() . . . 20
5.5 Tang::AstNodeCastFloat Class Reference . . . 21
5.5.1 Detailed Description . . . 23
5.5.2 Constructor & Destructor Documentation . . . 23
5.5.2.1 AstNodeCastFloat() . . . 23
5.5.3 Member Function Documentation . . . 23

5.5.3.1 makeCopy() . . . 23
5.6 Tang::AstNodeCastInteger Class Reference . . . 24
5.6.1 Detailed Description . . . 26
5.6.2 Constructor & Destructor Documentation . . . 26
5.6.2.1 AstNodeCastInteger() . . . 26
5.6.3 Member Function Documentation . . . 26
5.6.3.1 makeCopy() . . . 26
5.7 Tang::AstNodeFloat Class Reference . . . 27
5.7.1 Detailed Description . . . 29
5.7.2 Constructor & Destructor Documentation . . . 29
5.7.2.1 AstNodeFloat() . . . 29
5.7.3 Member Function Documentation . . . 29
5.7.3.1 makeCopy() . . . 29
5.8 Tang::AstNodeInteger Class Reference . . . 30
5.8.1 Detailed Description . . . 32
5.8.2 Constructor & Destructor Documentation . . . 32
5.8.2.1 AstNodeInteger() . . . 32
5.8.3 Member Function Documentation . . . 32
5.8.3.1 makeCopy() . . . 32
5.9 Tang::AstNodeNegative Class Reference . . . 33
5.9.1 Detailed Description . . . 35
5.9.2 Constructor & Destructor Documentation . . . 35
5.9.2.1 AstNodeNegative() . . . 35
5.9.3 Member Function Documentation . . . 35
5.9.3.1 makeCopy() . . . 35
5.10 Tang::ComputedExpression Class Reference . . . 36
5.10.1 Detailed Description . . . 37
5.10.2 Member Function Documentation . . . 37
5.10.2.1 __add() . . . 37
5.10.2.2 __boolean() . . . 37
5.10.2.3 __divide() . . . 38
5.10.2.4 __float() . . . 38
5.10.2.5 __integer() . . . 38
5.10.2.6 __modulo() . . . 38
5.10.2.7 __multiply() . . . 39
5.10.2.8 __negative() . . . 39
5.10.2.9 __subtract() . . . 39
5.10.2.10 dump() . . . 40
5.10.2.11 is_equal() [1/4] . . . 40
5.10.2.12 is_equal() [2/4] . . . 41
5.10.2.13 is_equal() [3/4] . . . 42
5.10.2.14 is_equal() [4/4] . . . 42

5.10.2.15 makeCopy() . . . 43
5.11 Tang::ComputedExpressionBoolean Class Reference . . . 43
5.11.1 Detailed Description . . . 44
5.11.2 Constructor & Destructor Documentation . . . 44
5.11.2.1 ComputedExpressionBoolean() . . . 44
5.11.3 Member Function Documentation . . . 45
5.11.3.1 __add() . . . 45
5.11.3.2 __boolean() . . . 45
5.11.3.3 __divide() . . . 45
5.11.3.4 __float() . . . 46
5.11.3.5 __integer() . . . 46
5.11.3.6 __modulo() . . . 46
5.11.3.7 __multiply() . . . 47
5.11.3.8 __negative() . . . 47
5.11.3.9 __subtract() . . . 47
5.11.3.10 dump() . . . 48
5.11.3.11 is_equal() [1/4] . . . 48
5.11.3.12 is_equal() [2/4] . . . 48
5.11.3.13 is_equal() [3/4] . . . 49
5.11.3.14 is_equal() [4/4] . . . 49
5.11.3.15 makeCopy() . . . 50
5.12 Tang::ComputedExpressionError Class Reference . . . 50
5.12.1 Detailed Description . . . 52
5.12.2 Constructor & Destructor Documentation . . . 52
5.12.2.1 ComputedExpressionError() . . . 52
5.12.3 Member Function Documentation . . . 52
5.12.3.1 __add() . . . 52
5.12.3.2 __boolean() . . . 52
5.12.3.3 __divide() . . . 53
5.12.3.4 __float() . . . 53
5.12.3.5 __integer() . . . 53
5.12.3.6 __modulo() . . . 54
5.12.3.7 __multiply() . . . 55
5.12.3.8 __negative() . . . 55
5.12.3.9 __subtract() . . . 55
5.12.3.10 dump() . . . 56
5.12.3.11 is_equal() [1/4] . . . 56
5.12.3.12 is_equal() [2/4] . . . 56
5.12.3.13 is_equal() [3/4] . . . 57
5.12.3.14 is_equal() [4/4] . . . 57
5.12.3.15 makeCopy() . . . 58
5.13 Tang::ComputedExpressionFloat Class Reference . . . 58

5.13.1 Detailed Description . . . 60
5.13.2 Constructor & Destructor Documentation . . . 60
5.13.2.1 ComputedExpressionFloat() . . . 60
5.13.3 Member Function Documentation . . . 60
5.13.3.1 __add() . . . 60
5.13.3.2 __boolean() . . . 61
5.13.3.3 __divide() . . . 61
5.13.3.4 __float() . . . 61
5.13.3.5 __integer() . . . 62
5.13.3.6 __modulo() . . . 62
5.13.3.7 __multiply() . . . 62
5.13.3.8 __negative() . . . 63
5.13.3.9 __subtract() . . . 63
5.13.3.10 dump() . . . 63
5.13.3.11 is_equal() [1/4] . . . 63
5.13.3.12 is_equal() [2/4] . . . 64
5.13.3.13 is_equal() [3/4] . . . 64
5.13.3.14 is_equal() [4/4] . . . 65
5.13.3.15 makeCopy() . . . 65
5.14 Tang::ComputedExpressionInteger Class Reference . . . 66
5.14.1 Detailed Description . . . 67
5.14.2 Constructor & Destructor Documentation . . . 67
5.14.2.1 ComputedExpressionInteger() . . . 67
5.14.3 Member Function Documentation . . . 68
5.14.3.1 __add() . . . 68
5.14.3.2 __boolean() . . . 68
5.14.3.3 __divide() . . . 68
5.14.3.4 __float() . . . 69
5.14.3.5 __integer() . . . 69
5.14.3.6 __modulo() . . . 69
5.14.3.7 __multiply() . . . 70
5.14.3.8 __negative() . . . 70
5.14.3.9 __subtract() . . . 70
5.14.3.10 dump() . . . 71
5.14.3.11 is_equal() [1/4] . . . 71
5.14.3.12 is_equal() [2/4] . . . 71
5.14.3.13 is_equal() [3/4] . . . 72
5.14.3.14 is_equal() [4/4] . . . 72
5.14.3.15 makeCopy() . . . 73
5.15 Tang::Error Class Reference . . . 73
5.15.1 Detailed Description . . . 75
5.15.2 Constructor & Destructor Documentation . . . 75

5.15.2.1 Error() [1/2] . . . 75
5.15.2.2 Error() [2/2] . . . 75
5.15.3 Friends And Related Function Documentation . . . 75
5.15.3.1 operator<< . . . 76
5.16 Tang::GarbageCollected Class Reference . . . 76
5.16.1 Detailed Description . . . 78
5.16.2 Constructor & Destructor Documentation . . . 78
5.16.2.1 GarbageCollected() [1/3] . . . 78
5.16.2.2 GarbageCollected() [2/3] . . . 78
5.16.2.3 ~GarbageCollected() . . . 79
5.16.2.4 GarbageCollected() [3/3] . . . 79
5.16.3 Member Function Documentation . . . 79
5.16.3.1 make() . . . 79
5.16.3.2 operator%() . . . 80
5.16.3.3 operator*() [1/2] . . . 80
5.16.3.4 operator*() [2/2] . . . 81
5.16.3.5 operator+() . . . 81
5.16.3.6 operator-() [1/2] . . . 82
5.16.3.7 operator-() [2/2] . . . 82
5.16.3.8 operator->() . . . 83
5.16.3.9 operator/() . . . 83
5.16.3.10 operator=() [1/2] . . . 84
5.16.3.11 operator=() [2/2] . . . 84
5.16.3.12 operator==() [1/4] . . . 85
5.16.3.13 operator==() [2/4] . . . 85
5.16.3.14 operator==() [3/4] . . . 86
5.16.3.15 operator==() [4/4] . . . 86
5.16.4 Friends And Related Function Documentation . . . 86
5.16.4.1 operator<< . . . 86
5.17 Tang::location Class Reference . . . 87
5.17.1 Detailed Description . . . 88
5.18 Tang::position Class Reference . . . 88
5.18.1 Detailed Description . . . 90
5.19 Tang::Program Class Reference . . . 90
5.19.1 Detailed Description . . . 91
5.19.2 Member Enumeration Documentation . . . 91
5.19.2.1 CodeType . . . 91
5.19.3 Constructor & Destructor Documentation . . . 92
5.19.3.1 Program() . . . 92
5.19.4 Member Function Documentation . . . 92
5.19.4.1 addBytecode() . . . 92
5.19.4.2 dumpBytecode() . . . 92

5.19.4.3 execute() . . . 93
5.19.4.4 getAst() . . . 93
5.19.4.5 getCode() . . . 93
5.19.4.6 getResult() . . . 94
5.20 Tang::SingletonObjectPool< T > Class Template Reference . . . 94
5.20.1 Detailed Description . . . 94
5.20.2 Member Function Documentation . . . 94
5.20.2.1 get() . . . 95
5.20.2.2 getInstance() . . . 95
5.20.2.3 recycle() . . . 95
5.21 Tang::TangBase Class Reference . . . 95
5.21.1 Detailed Description . . . 96
5.21.2 Constructor & Destructor Documentation . . . 96
5.21.2.1 TangBase() . . . 96
5.21.3 Member Function Documentation . . . 96
5.21.3.1 compileScript() . . . 96
5.22 Tang::TangScanner Class Reference . . . 97
5.22.1 Detailed Description . . . 98
5.22.2 Constructor & Destructor Documentation . . . 98
5.22.2.1 TangScanner() . . . 98
5.22.3 Member Function Documentation . . . 98
5.22.3.1 get_next_token() . . . 98

6 File Documentation **99**

6.1 build/generated/location.hh File Reference . . . 99
6.1.1 Detailed Description . . . 100
6.1.2 Function Documentation . . . 100
6.1.2.1 operator<<() [1/2] . . . 100
6.1.2.2 operator<<() [2/2] . . . 101
6.2 include/astNode.hpp File Reference . . . 101
6.2.1 Detailed Description . . . 102
6.3 include/astNodeBinary.hpp File Reference . . . 102
6.3.1 Detailed Description . . . 103
6.4 include/astNodeBoolean.hpp File Reference . . . 103
6.4.1 Detailed Description . . . 104
6.5 include/astNodeCastBoolean.hpp File Reference . . . 104
6.5.1 Detailed Description . . . 105
6.6 include/astNodeCastFloat.hpp File Reference . . . 105
6.6.1 Detailed Description . . . 106
6.7 include/astNodeCastInteger.hpp File Reference . . . 106
6.7.1 Detailed Description . . . 107
6.8 include/astNodeFloat.hpp File Reference . . . 107

6.8.1 Detailed Description . . . 108
6.9 include/astNodeInteger.hpp File Reference . . . 108
6.9.1 Detailed Description . . . 109
6.10 include/astNodeNegative.hpp File Reference . . . 109
6.10.1 Detailed Description . . . 110
6.11 include/computedExpression.hpp File Reference . . . 110
6.11.1 Detailed Description . . . 111
6.12 include/computedExpressionBoolean.hpp File Reference . . . 111
6.12.1 Detailed Description . . . 112
6.13 include/computedExpressionError.hpp File Reference . . . 112
6.13.1 Detailed Description . . . 113
6.14 include/computedExpressionFloat.hpp File Reference . . . 113
6.14.1 Detailed Description . . . 114
6.15 include/computedExpressionInteger.hpp File Reference . . . 114
6.15.1 Detailed Description . . . 115
6.16 include/error.hpp File Reference . . . 115
6.16.1 Detailed Description . . . 115
6.17 include/garbageCollected.hpp File Reference . . . 116
6.17.1 Detailed Description . . . 116
6.18 include/macros.hpp File Reference . . . 116
6.18.1 Detailed Description . . . 117
6.18.2 Macro Definition Documentation . . . 117
6.18.2.1 TANG_UNUSED . . . 117
6.19 include/opcode.hpp File Reference . . . 117
6.19.1 Detailed Description . . . 118
6.19.2 Enumeration Type Documentation . . . 118
6.19.2.1 Opcode . . . 118
6.20 include/program.hpp File Reference . . . 118
6.20.1 Detailed Description . . . 119
6.21 include/singletonObjectPool.hpp File Reference . . . 120
6.21.1 Detailed Description . . . 120
6.22 include/tang.hpp File Reference . . . 121
6.22.1 Detailed Description . . . 121
6.23 include/tangBase.hpp File Reference . . . 122
6.23.1 Detailed Description . . . 123
6.24 include/tangScanner.hpp File Reference . . . 123
6.24.1 Detailed Description . . . 124
6.25 src/astNode.cpp File Reference . . . 124
6.25.1 Detailed Description . . . 124
6.26 src/astNodeBinary.cpp File Reference . . . 125
6.26.1 Detailed Description . . . 125
6.27 src/astNodeBoolean.cpp File Reference . . . 125

6.27.1 Detailed Description . . . 126
6.28 src/astNodeCastBoolean.cpp File Reference . . . 126
6.28.1 Detailed Description . . . 127
6.29 src/astNodeCastFloat.cpp File Reference . . . 127
6.29.1 Detailed Description . . . 128
6.30 src/astNodeCastInteger.cpp File Reference . . . 128
6.30.1 Detailed Description . . . 129
6.31 src/astNodeFloat.cpp File Reference . . . 129
6.31.1 Detailed Description . . . 130
6.32 src/astNodeInteger.cpp File Reference . . . 130
6.32.1 Detailed Description . . . 131
6.33 src/astNodeNegative.cpp File Reference . . . 131
6.33.1 Detailed Description . . . 132
6.34 src/computedExpression.cpp File Reference . . . 132
6.34.1 Detailed Description . . . 133
6.35 src/computedExpressionBoolean.cpp File Reference . . . 133
6.35.1 Detailed Description . . . 134
6.36 src/computedExpressionError.cpp File Reference . . . 134
6.36.1 Detailed Description . . . 134
6.37 src/computedExpressionFloat.cpp File Reference . . . 134
6.37.1 Detailed Description . . . 135
6.38 src/computedExpressionInteger.cpp File Reference . . . 135
6.38.1 Detailed Description . . . 136
6.39 src/error.cpp File Reference . . . 136
6.39.1 Detailed Description . . . 136
6.39.2 Function Documentation . . . 136
6.39.2.1 operator<<() . . . 136
6.40 src/program-dumpBytecode.cpp File Reference . . . 137
6.40.1 Detailed Description . . . 137
6.40.2 Macro Definition Documentation . . . 138
6.40.2.1 DUMPPROGRAMCHECK . . . 138
6.41 src/program-execute.cpp File Reference . . . 138
6.41.1 Detailed Description . . . 139
6.41.2 Macro Definition Documentation . . . 139
6.41.2.1 EXECUTEPROGRAMCHECK . . . 139
6.41.2.2 STACKCHECK . . . 139
6.42 src/program.cpp File Reference . . . 140
6.42.1 Detailed Description . . . 140
6.43 src/tangBase.cpp File Reference . . . 140
6.43.1 Detailed Description . . . 141
6.44 test/test.cpp File Reference . . . 141
6.44.1 Detailed Description . . . 142

6.45 test/testGarbageCollected.cpp File Reference 142
6.45.1 Detailed Description 143
6.46 test/testSingletonObjectPool.cpp File Reference 143
6.46.1 Detailed Description 144

Index **145**

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 . . . 9
- Tang::AstNodeBinary . . . 12
- Tang::AstNodeBoolean . . . 15
- Tang::AstNodeCastBoolean . . . 18
- Tang::AstNodeCastFloat . . . 21
- Tang::AstNodeCastInteger . . . 24
- Tang::AstNodeFloat . . . 27
- Tang::AstNodeInteger . . . 30
- Tang::AstNodeNegative . . . 33

Tang::ComputedExpression . . . 36
- Tang::ComputedExpressionBoolean . . . 43
- Tang::ComputedExpressionError . . . 50
- Tang::ComputedExpressionFloat . . . 58
- Tang::ComputedExpressionInteger . . . 66

Tang::Error . . . 73
Tang::GarbageCollected . . . 76
Tang::location . . . 87
Tang::position . . . 88
Tang::Program . . . 90
Tang::SingletonObjectPool< T > . . . 94
Tang::TangBase . . . 95
TangTangFlexLexer
- Tang::TangScanner . . . 97

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) . . . 9

Tang::AstNodeBinary
: An AstNode that represents a binary expression . . . 12

Tang::AstNodeBoolean
: An AstNode that represents a boolean literal . . . 15

Tang::AstNodeCastBoolean
: An AstNode that represents a typecast to a boolean . . . 18

Tang::AstNodeCastFloat
: An AstNode that represents a typecast to a float . . . 21

Tang::AstNodeCastInteger
: An AstNode that represents a typecast to an integer . . . 24

Tang::AstNodeFloat
: An AstNode that represents an float literal . . . 27

Tang::AstNodeInteger
: An AstNode that represents an integer literal . . . 30

Tang::AstNodeNegative
: An AstNode that represents a unary negation . . . 33

Tang::ComputedExpression
: Represents the result of a computation that has been executed . . . 36

Tang::ComputedExpressionBoolean
: Represents an Boolean that is the result of a computation . . . 43

Tang::ComputedExpressionError
: Represents a Runtime Error . . . 50

Tang::ComputedExpressionFloat
: Represents a Float that is the result of a computation . . . 58

Tang::ComputedExpressionInteger
: Represents an Integer that is the result of a computation . . . 66

Tang::Error
: Used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error . . . 73

Tang::GarbageCollected
: A container that acts as a resource-counting garbage collector for the specified type . . . 76

Tang::location
: Two points in a source file . . . 87

Tang::position
A point in a source file 88
Tang::Program
Represents a compiled script or template that may be executed 90
Tang::SingletonObjectPool< T >
A thread-safe, singleton object pool of the designated type 94
Tang::TangBase
The base class for the Tang programming language 95
Tang::TangScanner
The Flex lexer class for the main Tang language 97

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 . . . 99
include/astNode.hpp
Declare the Tang::AstNode base class . . . 101
include/astNodeBinary.hpp
Declare the Tang::AstNodeBinary class . . . 102
include/astNodeBoolean.hpp
Declare the Tang::AstNodeBoolean class . . . 103
include/astNodeCastBoolean.hpp
Declare the Tang::AstNodeBoolean class . . . 104
include/astNodeCastFloat.hpp
Declare the Tang::AstNodeFloat class . . . 105
include/astNodeCastInteger.hpp
Declare the Tang::AstNodeInteger class . . . 106
include/astNodeFloat.hpp
Declare the Tang::AstNodeFloat class . . . 107
include/astNodeInteger.hpp
Declare the Tang::AstNodeInteger class . . . 108
include/astNodeNegative.hpp
Declare the Tang::AstNodeNegative class . . . 109
include/computedExpression.hpp
Declare the Tang::ComputedExpression base class . . . 110
include/computedExpressionBoolean.hpp
Declare the Tang::ComputedExpressionBoolean class . . . 111
include/computedExpressionError.hpp
Declare the Tang::ComputedExpressionError class . . . 112
include/computedExpressionFloat.hpp
Declare the Tang::ComputedExpressionFloat class . . . 113
include/computedExpressionInteger.hpp
Declare the Tang::ComputedExpressionInteger class . . . 114
include/error.hpp
Declare the Tang::Error class used to describe syntax and runtime errors . . . 115
include/garbageCollected.hpp
Declare the Tang::GarbageCollected class . . . 116
include/macros.hpp
Contains generic macros . . . 116

include/opcode.hpp
Declare the Opcodes used in the Bytecode representation of a program . . . 117
include/program.hpp
Declare the Tang::Program class used to compile and execute source code . . . 118
include/singletonObjectPool.hpp
Declare the Tang::SingletonObjectPool class . . . 120
include/tang.hpp
Header file supplied for use by 3rd party code so that they can easily include all necessary headers . . . 121
include/tangBase.hpp
Declare the Tang::TangBase class used to interact with Tang . . . 122
include/tangScanner.hpp
Declare the Tang::TangScanner used to tokenize a Tang script . . . 123
src/astNode.cpp
Define the Tang::AstNode class . . . 124
src/astNodeBinary.cpp
Define the Tang::AstNodeBinary class . . . 125
src/astNodeBoolean.cpp
Define the Tang::AstNodeBoolean class . . . 125
src/astNodeCastBoolean.cpp
Define the Tang::AstNodeCastBoolean class . . . 126
src/astNodeCastFloat.cpp
Define the Tang::shared_ptr<AstNode>CastFloat class . . . 127
src/astNodeCastInteger.cpp
Define the Tang::AstNodeCastInteger class . . . 128
src/astNodeFloat.cpp
Define the Tang::AstNodeFloat class . . . 129
src/astNodeInteger.cpp
Define the Tang::AstNodeInteger class . . . 130
src/astNodeNegative.cpp
Define the Tang::AstNodeNegative class . . . 131
src/computedExpression.cpp
Define the Tang::ComputedExpression class . . . 132
src/computedExpressionBoolean.cpp
Define the Tang::ComputedExpressionBoolean class . . . 133
src/computedExpressionError.cpp
Define the Tang::ComputedExpressionError class . . . 134
src/computedExpressionFloat.cpp
Define the Tang::ComputedExpressionFloat class . . . 134
src/computedExpressionInteger.cpp
Define the Tang::ComputedExpressionInteger class . . . 135
src/error.cpp
Define the Tang::Error class . . . 136
src/program-dumpBytecode.cpp
Define the Tang::Program::dumpBytecode method . . . 137
src/program-execute.cpp
Define the Tang::Program::execute method . . . 138
src/program.cpp
Define the Tang::Program class . . . 140
src/tangBase.cpp
Define the Tang::TangBase class . . . 140
test/test.cpp
Test the general language behaviors . . . 141
test/testGarbageCollected.cpp
Test the generic behavior of the Tang::GarbageCollected class . . . 142
test/testSingletonObjectPool.cpp
Test the generic behavior of the Tang::SingletonObjectPool class . . . 143

Chapter 5

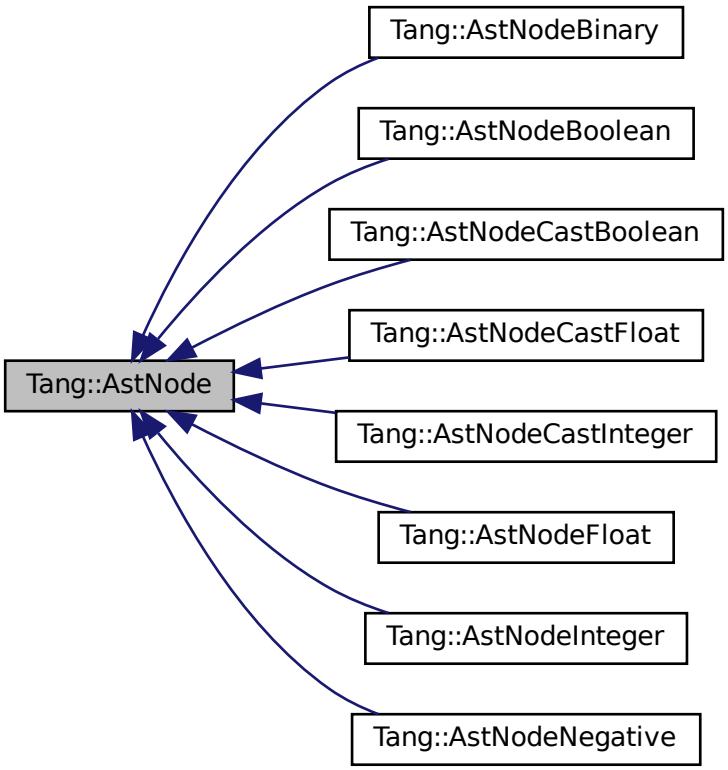
Class Documentation

5.1 Tang::AstNode Class Reference

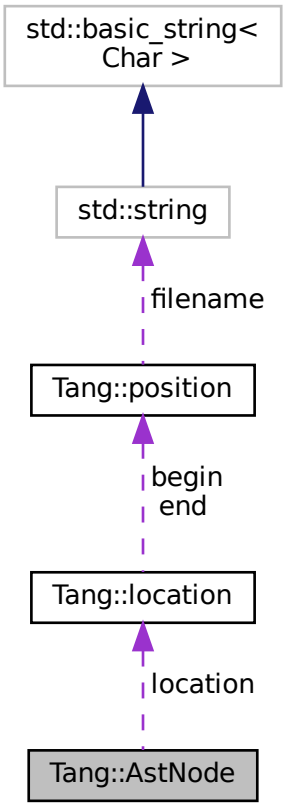
Base class for representing nodes of an Abstract Syntax Tree (AST).

`#include <astNode.hpp>`

Inheritance diagram for Tang::AstNode:



Collaboration diagram for Tang::AstNode:



Public Member Functions

- virtual ~AstNode ()

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

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

  Compile the ast of the provided Tang::Program.
- virtual std::shared_ptr< AstNode > makeCopy () const

  Provide a copy of the AstNode (recursively, if appropriate).

Protected Member Functions

- AstNode (Tang::location location)

  The generic constructor.

Protected Attributes

- Tang::location location

  The location associated with this node.

5.1.1 Detailed Description

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

There will be *many* derived classes, each one conveying the syntactic meaning of the code that it represents.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AstNode()

```
AstNode::AstNode (
            Tang::location location )  [protected]
```

The generic constructor.

It should never be called on its own.

Parameters

location	The location associated with this node.

5.1.3 Member Function Documentation

5.1.3.1 makeCopy()

```
shared_ptr< AstNode > AstNode::makeCopy ( ) const  [virtual]
```

Provide a copy of the AstNode (recursively, if appropriate).

Returns

A pointer to a new AstNode that is a copy of the current AstNode.

Reimplemented in Tang::AstNodeNegative, Tang::AstNodeInteger, Tang::AstNodeFloat, Tang::AstNodeCastInteger, Tang::AstNodeCastFloat, Tang::AstNodeCastBoolean, Tang::AstNodeBoolean, and Tang::AstNodeBinary.

Here is the call graph for this function:

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

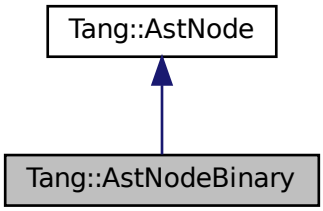
- include/astNode.hpp
- src/astNode.cpp

5.2 Tang::AstNodeBinary Class Reference

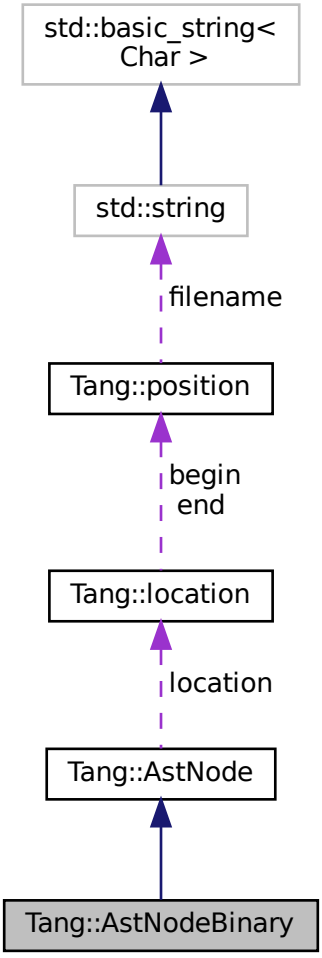
An AstNode that represents a binary expression.

`#include <astNodeBinary.hpp>`

Inheritance diagram for Tang::AstNodeBinary:



Collaboration diagram for Tang::AstNodeBinary:



Public Types

- enum **Operation** {
  Add , **Subtract** , **Multiply** , **Divide** ,
  Modulo }

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 shared_ptr< AstNode > makeCopy () const override

  Provide a copy of the AstNode (recursively, if appropriate).

Protected Attributes

- Tang::location location

  The location associated with this node.

5.2.1 Detailed Description

An AstNode that represents a binary expression.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AstNodeBinary()

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

The constructor.

Parameters

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

5.2.3 Member Function Documentation

5.2.3.1 makeCopy()

```
shared_ptr< AstNode > AstNodeBinary::makeCopy ( ) const  [override], [virtual]
```

Provide a copy of the AstNode (recursively, if appropriate).

Returns

A pointer to a new AstNode that is a copy of the current AstNode.

Reimplemented from Tang::AstNode.

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

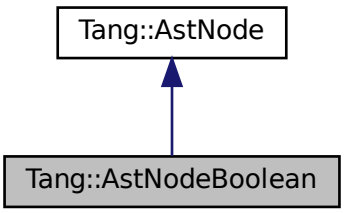
- include/astNodeBinary.hpp
- src/astNodeBinary.cpp

5.3 Tang::AstNodeBoolean Class Reference

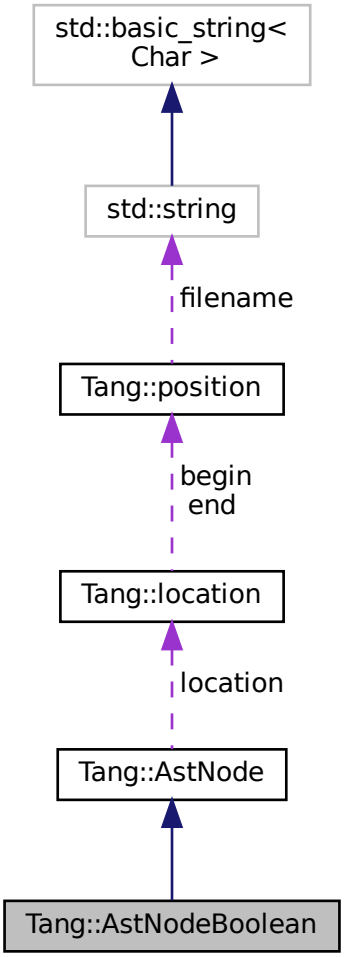
An AstNode that represents a boolean literal.

`#include <astNodeBoolean.hpp>`

Inheritance diagram for Tang::AstNodeBoolean:



Collaboration diagram for Tang::AstNodeBoolean:



Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual shared_ptr< AstNode > makeCopy () const override

  Provide a copy of the AstNode (recursively, if appropriate).

Protected Attributes

- Tang::location location

  The location associated with this node.

5.3.1 Detailed Description

An AstNode that represents a boolean literal.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 AstNodeBoolean()

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

The constructor.

Parameters

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

5.3.3 Member Function Documentation

5.3.3.1 makeCopy()

```
shared_ptr< AstNode > AstNodeBoolean::makeCopy ( ) const  [override], [virtual]
```

Provide a copy of the AstNode (recursively, if appropriate).

Returns

A pointer to a new AstNode that is a copy of the current AstNode.

Reimplemented from Tang::AstNode.

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

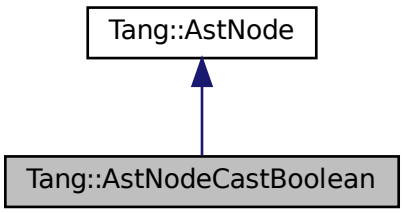
- include/astNodeBoolean.hpp
- src/astNodeBoolean.cpp

5.4 Tang::AstNodeCastBoolean Class Reference

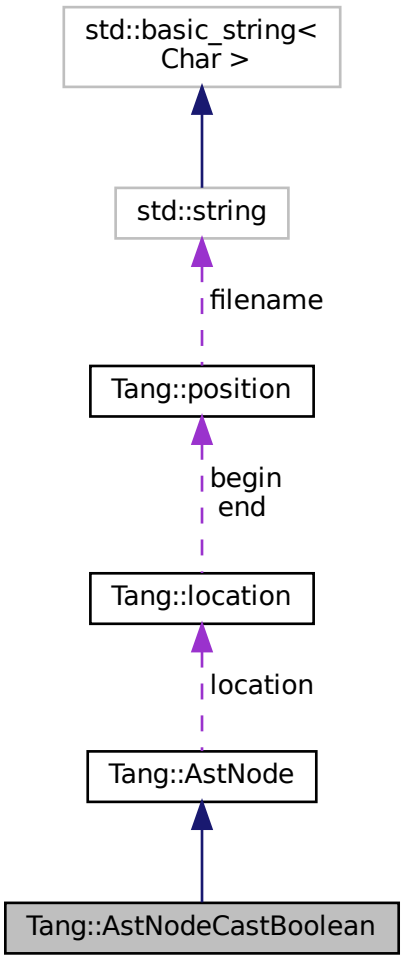
An AstNode that represents a typecast to a boolean.

`#include <astNodeCastBoolean.hpp>`

Inheritance diagram for Tang::AstNodeCastBoolean:



Collaboration diagram for Tang::AstNodeCastBoolean:



Public Member Functions

- AstNodeCastBoolean (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 shared_ptr< AstNode > makeCopy () const override

  Provide a copy of the AstNode (recursively, if appropriate).

Protected Attributes

- Tang::location location

  The location associated with this node.

5.4.1 Detailed Description

An AstNode that represents a typecast to a boolean.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 AstNodeCastBoolean()

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

The constructor.

Parameters

expression	The expression to be typecast.
location	The location associated with this node.

5.4.3 Member Function Documentation

5.4.3.1 makeCopy()

```
shared_ptr< AstNode > AstNodeCastBoolean::makeCopy ( ) const  [override], [virtual]
```

Provide a copy of the AstNode (recursively, if appropriate).

Returns

A pointer to a new AstNode that is a copy of the current AstNode.

Reimplemented from Tang::AstNode.

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

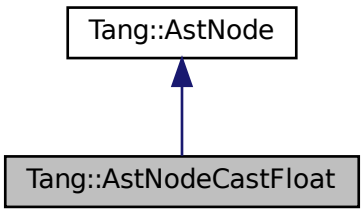
- include/astNodeCastBoolean.hpp
- src/astNodeCastBoolean.cpp

5.5 Tang::AstNodeCastFloat Class Reference

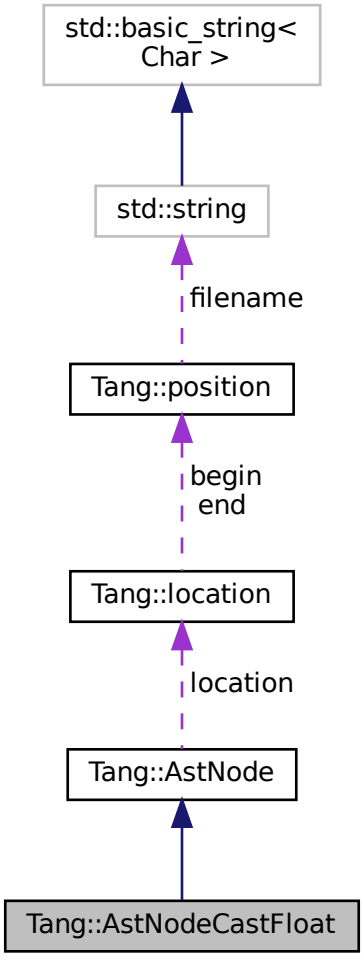
An AstNode that represents a typecast to a float.

`#include <astNodeCastFloat.hpp>`

Inheritance diagram for Tang::AstNodeCastFloat:



Collaboration diagram for Tang::AstNodeCastFloat:



Public Member Functions

- AstNodeCastFloat (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 shared_ptr< AstNode > makeCopy () const override

  Provide a copy of the AstNode (recursively, if appropriate).

Protected Attributes

- Tang::location location

  The location associated with this node.

5.5.1 Detailed Description

An AstNode that represents a typecast to a float.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 AstNodeCastFloat()

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

The constructor.

Parameters

expression	The expression to be typecast.
location	The location associated with this node.

5.5.3 Member Function Documentation

5.5.3.1 makeCopy()

```
shared_ptr< AstNode > AstNodeCastFloat::makeCopy ( ) const  [override], [virtual]
```

Provide a copy of the AstNode (recursively, if appropriate).

Returns

A pointer to a new AstNode that is a copy of the current AstNode.

Reimplemented from Tang::AstNode.

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

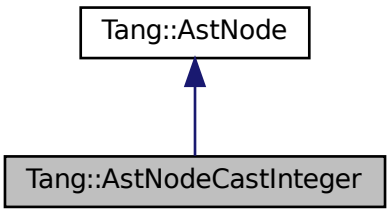
- include/astNodeCastFloat.hpp
- src/astNodeCastFloat.cpp

5.6 Tang::AstNodeCastInteger Class Reference

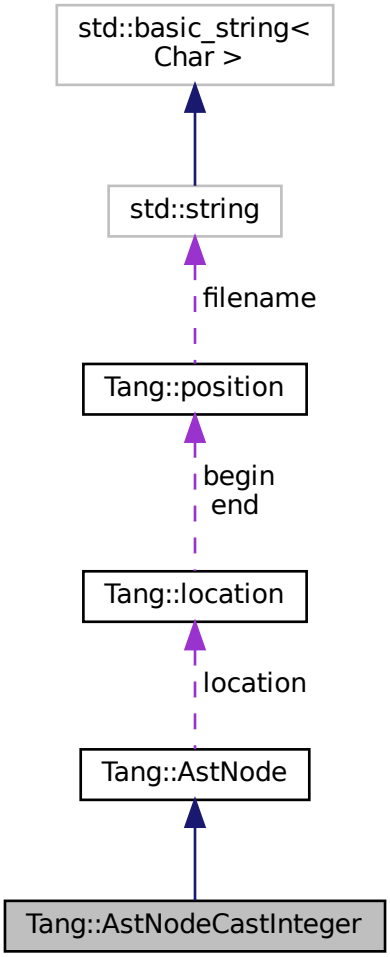
An AstNode that represents a typecast to an integer.

`#include <astNodeCastInteger.hpp>`

Inheritance diagram for Tang::AstNodeCastInteger:



Collaboration diagram for Tang::AstNodeCastInteger:



Public Member Functions

- AstNodeCastInteger (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 shared_ptr< AstNode > makeCopy () const override

  Provide a copy of the AstNode (recursively, if appropriate).

Protected Attributes

- Tang::location location

  The location associated with this node.

5.6.1 Detailed Description

An AstNode that represents a typecast to an integer.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 AstNodeCastInteger()

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

The constructor.

Parameters

expression	The expression to be typecast.
location	The location associated with this node.

5.6.3 Member Function Documentation

5.6.3.1 makeCopy()

```
shared_ptr< AstNode > AstNodeCastInteger::makeCopy ( ) const  [override], [virtual]
```

Provide a copy of the AstNode (recursively, if appropriate).

Returns

A pointer to a new AstNode that is a copy of the current AstNode.

Reimplemented from Tang::AstNode.

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

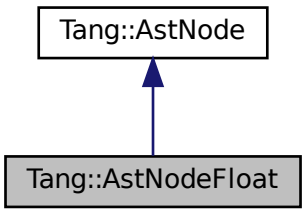
- include/astNodeCastInteger.hpp
- src/astNodeCastInteger.cpp

5.7 Tang::AstNodeFloat Class Reference

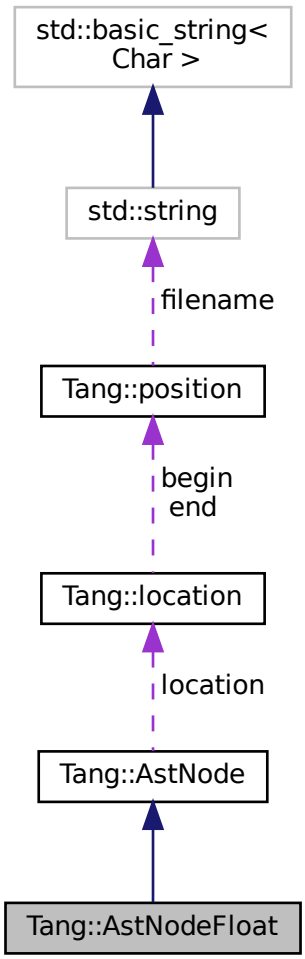
An AstNode that represents an float literal.

`#include <astNodeFloat.hpp>`

Inheritance diagram for Tang::AstNodeFloat:



Collaboration diagram for Tang::AstNodeFloat:



Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual shared_ptr< AstNode > makeCopy () const override

  Provide a copy of the AstNode (recursively, if appropriate).

Protected Attributes

- Tang::location location

  The location associated with this node.

5.7.1 Detailed Description

An AstNode that represents an float literal.

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

5.7.2 Constructor & Destructor Documentation

5.7.2.1 AstNodeFloat()

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

The constructor.

Parameters

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

5.7.3 Member Function Documentation

5.7.3.1 makeCopy()

```
shared_ptr< AstNode > AstNodeFloat::makeCopy ( ) const  [override], [virtual]
```

Provide a copy of the AstNode (recursively, if appropriate).

Returns

A pointer to a new AstNode that is a copy of the current AstNode.

Reimplemented from Tang::AstNode.

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

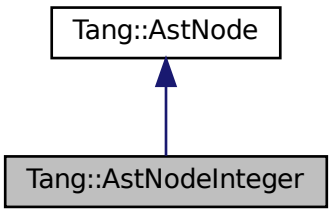
- include/astNodeFloat.hpp
- src/astNodeFloat.cpp

5.8 Tang::AstNodeInteger Class Reference

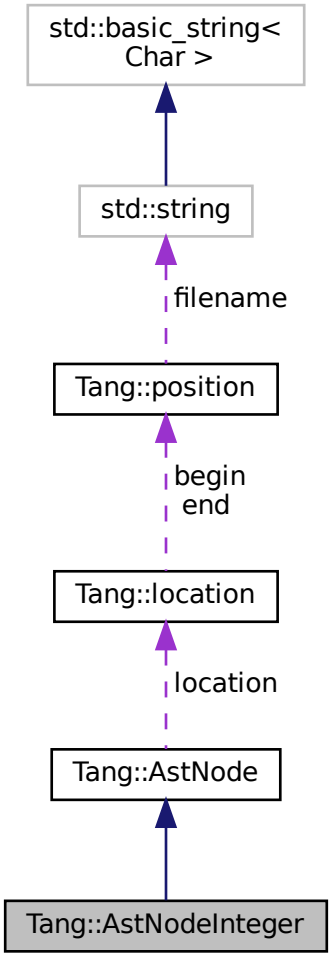
An AstNode that represents an integer literal.

`#include <astNodeInteger.hpp>`

Inheritance diagram for Tang::AstNodeInteger:



Collaboration diagram for Tang::AstNodeInteger:



Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual shared_ptr< AstNode > makeCopy () const override

  Provide a copy of the AstNode (recursively, if appropriate).

Protected Attributes

- Tang::location location

  The location associated with this node.

5.8.1 Detailed Description

An AstNode that represents an integer literal.

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

5.8.2 Constructor & Destructor Documentation

5.8.2.1 AstNodeInteger()

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

The constructor.

Parameters

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

5.8.3 Member Function Documentation

5.8.3.1 makeCopy()

```
shared_ptr< AstNode > AstNodeInteger::makeCopy ( ) const  [override], [virtual]
```

Provide a copy of the AstNode (recursively, if appropriate).

Returns

A pointer to a new AstNode that is a copy of the current AstNode.

Reimplemented from Tang::AstNode.

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

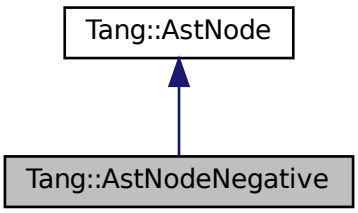
- include/astNodeInteger.hpp
- src/astNodeInteger.cpp

5.9 Tang::AstNodeNegative Class Reference

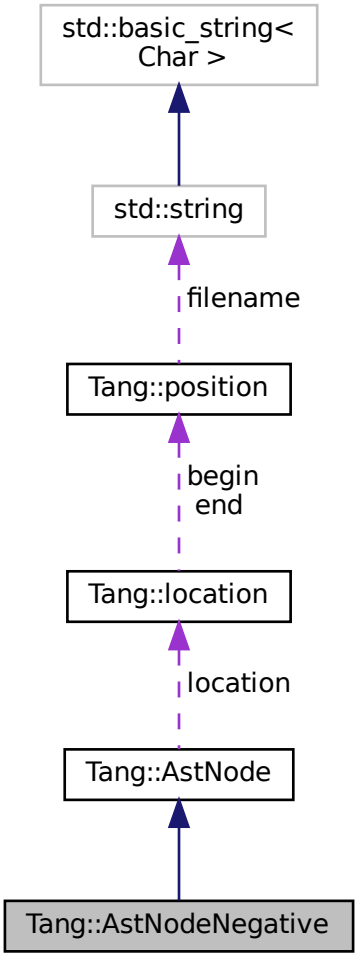
An AstNode that represents a unary negation.

`#include <astNodeNegative.hpp>`

Inheritance diagram for Tang::AstNodeNegative:



Collaboration diagram for Tang::AstNodeNegative:



Public Member Functions

- AstNodeNegative (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 shared_ptr< AstNode > makeCopy () const override

  Provide a copy of the AstNode (recursively, if appropriate).

Protected Attributes

- Tang::location location

  The location associated with this node.

5.9.1 Detailed Description

An AstNode that represents a unary negation.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 AstNodeNegative()

```
AstNodeNegative::AstNodeNegative (
            shared_ptr< AstNode > operand,
            Tang::location location )
```

The constructor.

Parameters

operand	The expression to negate.
location	The location associated with the expression.

5.9.3 Member Function Documentation

5.9.3.1 makeCopy()

```
shared_ptr< AstNode > AstNodeNegative::makeCopy ( ) const  [override], [virtual]
```

Provide a copy of the AstNode (recursively, if appropriate).

Returns

A pointer to a new AstNode that is a copy of the current AstNode.

Reimplemented from Tang::AstNode.

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

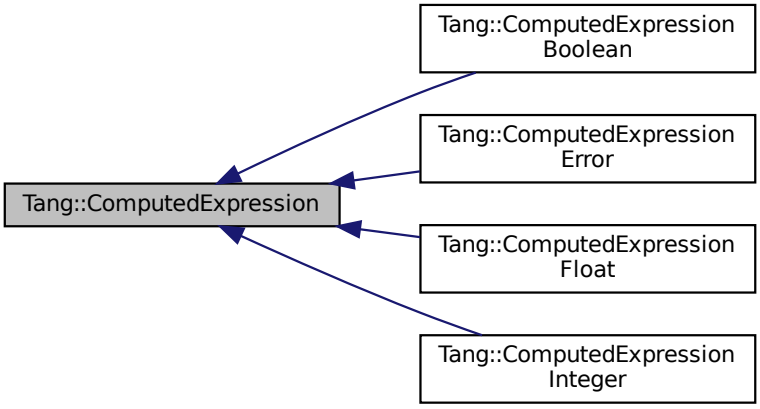
- include/astNodeNegative.hpp
- src/astNodeNegative.cpp

5.10 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 ComputedExpression * makeCopy () const

  Make a copy of the ComputedExpression (recursively, if appropriate).
- virtual bool is_equal (const int &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const double &val) const

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

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Error &val) const

  Check whether or not the computed expression is equal to another value.
- virtual 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 __integer () const

  Perform a type cast to integer.

- virtual GarbageCollected __float () const

  Perform a type cast to float.

- virtual GarbageCollected __boolean () const

  Perform a type cast to boolean.

5.10.1 Detailed Description

Represents the result of a computation that has been executed.

5.10.2 Member Function Documentation

5.10.2.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

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

5.10.2.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

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

5.10.2.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

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

5.10.2.4 __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, and Tang::ComputedExpressionBoolean.

5.10.2.5 __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, and Tang::ComputedExpressionBoolean.

5.10.2.6 __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.

5.10.2.7 __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, and Tang::ComputedExpressionFloat.

5.10.2.8 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

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

5.10.2.9 __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, and Tang::ComputedExpressionFloat.

5.10.2.10 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::ComputedExpressionInteger, Tang::ComputedExpressionFloat, Tang::ComputedExpressionError, and Tang::ComputedExpressionBoolean.

5.10.2.11 is_equal() [1/4]

```
virtual bool Tang::ComputedExpression::is_equal (
            const bool & val ) const  [virtual]
```

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionBoolean.

5.10.2.12 is_equal() [2/4]

```
virtual bool Tang::ComputedExpression::is_equal (
            const double & val ) const  [virtual]
```

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.10.2.13 is_equal() [3/4]

```
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.

5.10.2.14 is_equal() [4/4]

```
virtual bool Tang::ComputedExpression::is_equal (
            const int & val ) const  [virtual]
```

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.10.2.15 makeCopy()

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

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

Returns

A pointer to the new ComputedExpression.

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

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

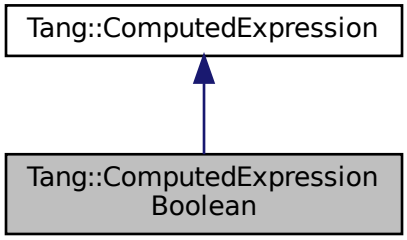
- include/computedExpression.hpp
- src/computedExpression.cpp

5.11 Tang::ComputedExpressionBoolean Class Reference

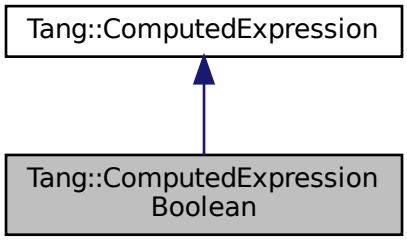
Represents an Boolean that is the result of a computation.

`#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.
- ComputedExpression * 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 __integer () const override

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

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

  Perform a type cast to boolean.
- virtual bool is_equal (const int &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const double &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Error &val) const

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

5.11.1 Detailed Description

Represents an Boolean that is the result of a computation.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 ComputedExpressionBoolean()

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

Construct an Boolean result.

Parameters

val	The boolean value.

5.11.3 Member Function Documentation

5.11.3.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

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

5.11.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.11.3.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

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

5.11.3.4 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.11.3.5 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.11.3.6 __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.

5.11.3.7 __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, and Tang::ComputedExpressionFloat.

5.11.3.8 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

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

5.11.3.9 __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, and Tang::ComputedExpressionFloat.

5.11.3.10 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.

5.11.3.11 is_equal() [1/4]

```
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.

5.11.3.12 is_equal() [2/4]

```
virtual bool Tang::ComputedExpression::is_equal (
            const double & val ) const  [virtual], [inherited]
```

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.11.3.13 is_equal() [3/4]

```
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.

5.11.3.14 is_equal() [4/4]

```
virtual bool Tang::ComputedExpression::is_equal (
            const int & val ) const  [virtual], [inherited]
```

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.11.3.15 makeCopy()

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

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

Returns

A pointer to the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:

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

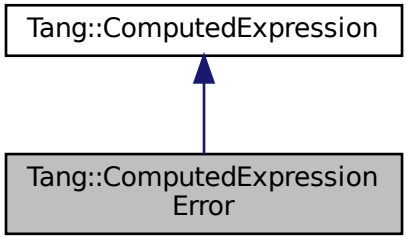
- include/computedExpressionBoolean.hpp
- src/computedExpressionBoolean.cpp

5.12 Tang::ComputedExpressionError Class Reference

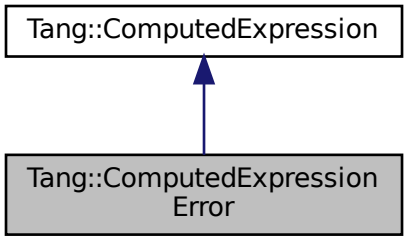
Represents a Runtime Error.

```
#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.
- ComputedExpression * 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 bool is_equal (const int &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const double &val) const

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

  Check whether or not the computed expression is equal to another value.
- virtual 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 __integer () const

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

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

  Perform a type cast to boolean.

5.12.1 Detailed Description

Represents a Runtime Error.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 ComputedExpressionError()

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

Construct a Runtime Error.

Parameters

error	The Tang::Error object.

5.12.3 Member Function Documentation

5.12.3.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

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

5.12.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

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

5.12.3.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

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

5.12.3.4 __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, and Tang::ComputedExpressionBoolean.

5.12.3.5 __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, and Tang::ComputedExpressionBoolean.

5.12.3.6 __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.

5.12.3.7 __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, and Tang::ComputedExpressionFloat.

5.12.3.8 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

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

5.12.3.9 __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, and Tang::ComputedExpressionFloat.

5.12.3.10 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.

5.12.3.11 is_equal() [1/4]

```
virtual bool Tang::ComputedExpression::is_equal (
            const bool & val ) const  [virtual], [inherited]
```

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionBoolean.

5.12.3.12 is_equal() [2/4]

```
virtual bool Tang::ComputedExpression::is_equal (
            const double & val ) const  [virtual], [inherited]
```

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.12.3.13 is_equal() [3/4]

```
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.

5.12.3.14 is_equal() [4/4]

```
virtual bool Tang::ComputedExpression::is_equal (
            const int & val ) const  [virtual], [inherited]
```

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.12.3.15 makeCopy()

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

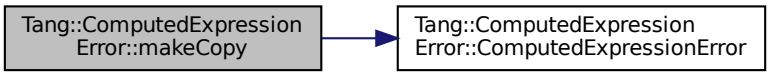
Make a copy of the ComputedExpression (recursively, if appropriate).

Returns

A pointer to the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



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

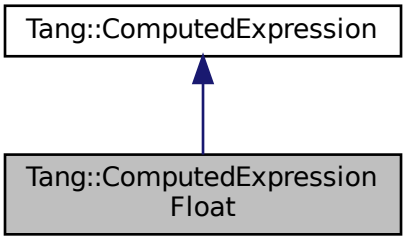
- include/computedExpressionError.hpp
- src/computedExpressionError.cpp

5.13 Tang::ComputedExpressionFloat Class Reference

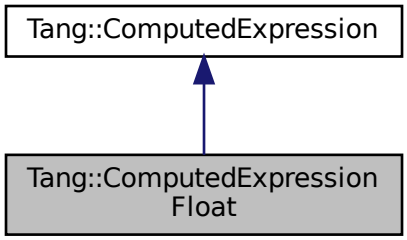
Represents a Float that is the result of a computation.

```
#include <computedExpressionFloat.hpp>
```

Inheritance diagram for Tang::ComputedExpressionFloat:



Collaboration diagram for Tang::ComputedExpressionFloat:



Public Member Functions

- ComputedExpressionFloat (double val)

  Construct a Float result.
- virtual std::string dump () const override

  Output the contents of the ComputedExpression as a string.
- ComputedExpression * makeCopy () const override

  Make a copy of the ComputedExpression (recursively, if appropriate).
- virtual bool is_equal (const int &val) const override

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const double &val) const override

  Check whether or not the computed expression is equal to another value.
- virtual GarbageCollected __add (const GarbageCollected &rhs) const override

  Compute the result of adding this value and the supplied value.
- virtual GarbageCollected __subtract (const GarbageCollected &rhs) const override

  Compute the result of subtracting this value and the supplied value.
- virtual GarbageCollected __multiply (const GarbageCollected &rhs) const override

  Compute the result of multiplying this value and the supplied value.
- virtual GarbageCollected __divide (const GarbageCollected &rhs) const override

  Compute the result of dividing this value and the supplied value.
- virtual GarbageCollected __negative () const override

  Compute the result of negating this value.
- virtual GarbageCollected __integer () const override

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

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

  Perform a type cast to boolean.
- virtual bool is_equal (const bool &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Error &val) const

  Check whether or not the computed expression is equal to another value.
- virtual GarbageCollected __modulo (const GarbageCollected &rhs) const

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

Friends

- class **ComputedExpressionInteger**

5.13.1 Detailed Description

Represents a Float that is the result of a computation.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 ComputedExpressionFloat()

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

Construct a Float result.

Parameters

val	The float value.

5.13.3 Member Function Documentation

5.13.3.1 __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.

5.13.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.13.3.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.13.3.4 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.13.3.5 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.13.3.6 __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.

5.13.3.7 __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.

5.13.3.8 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.13.3.9 __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.

5.13.3.10 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.

5.13.3.11 is_equal() [1/4]

```
virtual bool Tang::ComputedExpression::is_equal (
            const bool & val ) const  [virtual], [inherited]
```

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionBoolean.

5.13.3.12 is_equal() [2/4]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.13.3.13 is_equal() [3/4]

```
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.

5.13.3.14 is_equal() [4/4]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.13.3.15 makeCopy()

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

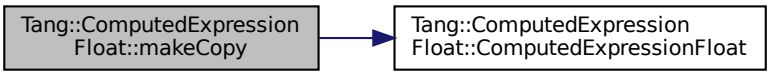
Make a copy of the ComputedExpression (recursively, if appropriate).

Returns

A pointer to the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



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

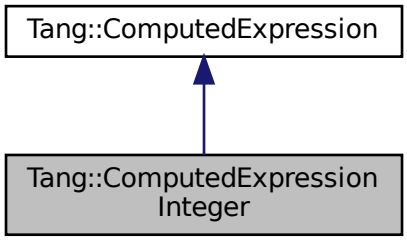
- include/computedExpressionFloat.hpp
- src/computedExpressionFloat.cpp

5.14 Tang::ComputedExpressionInteger Class Reference

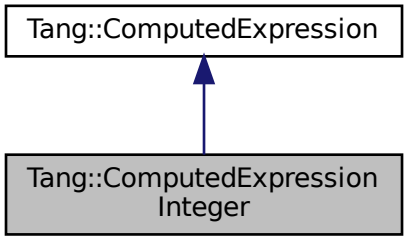
Represents an Integer that is the result of a computation.

`#include <computedExpressionInteger.hpp>`

Inheritance diagram for Tang::ComputedExpressionInteger:



Collaboration diagram for Tang::ComputedExpressionInteger:



Public Member Functions

- ComputedExpressionInteger (int64_t val)

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

  Output the contents of the ComputedExpression as a string.
- ComputedExpression * makeCopy () const override

  Make a copy of the ComputedExpression (recursively, if appropriate).
- virtual bool is_equal (const int &val) const override

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const double &val) const override

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

- virtual GarbageCollected __add (const GarbageCollected &rhs) const override

  Compute the result of adding this value and the supplied value.
- virtual GarbageCollected __subtract (const GarbageCollected &rhs) const override

  Compute the result of subtracting this value and the supplied value.
- virtual GarbageCollected __multiply (const GarbageCollected &rhs) const override

  Compute the result of multiplying this value and the supplied value.
- virtual GarbageCollected __divide (const GarbageCollected &rhs) const override

  Compute the result of dividing this value and the supplied value.
- virtual GarbageCollected __modulo (const GarbageCollected &rhs) const override

  Compute the result of moduloing this value and the supplied value.
- virtual GarbageCollected __negative () const override

  Compute the result of negating this value.
- virtual GarbageCollected __integer () const override

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

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

  Perform a type cast to boolean.
- virtual bool is_equal (const bool &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Error &val) const

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

Friends

- class **ComputedExpressionFloat**

5.14.1 Detailed Description

Represents an Integer that is the result of a computation.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 ComputedExpressionInteger()

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

Construct an Integer result.

Parameters

val	The integer value.

5.14.3 Member Function Documentation

5.14.3.1 __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.

5.14.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.14.3.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.14.3.4 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.14.3.5 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.14.3.6 __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.

5.14.3.7 __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.

5.14.3.8 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.14.3.9 __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.

5.14.3.10 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.

5.14.3.11 is_equal() [1/4]

```
virtual bool Tang::ComputedExpression::is_equal (
            const bool & val ) const  [virtual], [inherited]
```

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionBoolean.

5.14.3.12 is_equal() [2/4]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.14.3.13 is_equal() [3/4]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Error & val ) const  [virtual], [inherited]
```

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionError.

5.14.3.14 is_equal() [4/4]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.14.3.15 makeCopy()

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

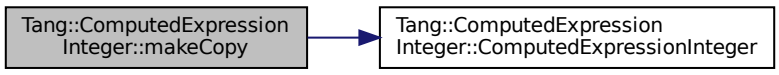
Make a copy of the ComputedExpression (recursively, if appropriate).

Returns

A pointer to the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



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

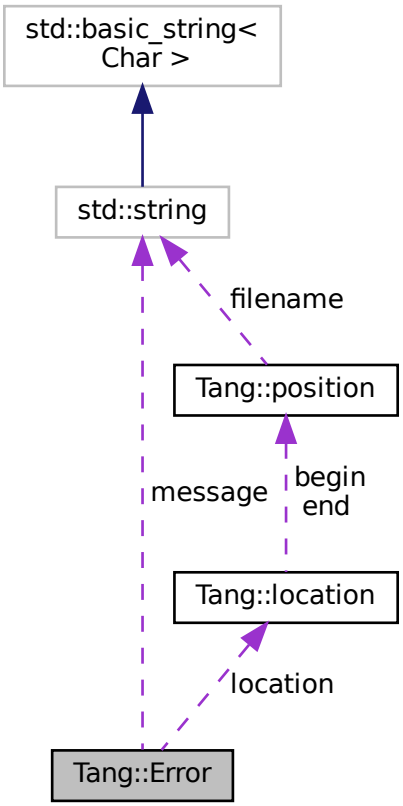
- include/computedExpressionInteger.hpp
- src/computedExpressionInteger.cpp

5.15 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.

5.15.1 Detailed Description

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

5.15.2 Constructor & Destructor Documentation

5.15.2.1 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.

5.15.2.2 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.

5.15.3 Friends And Related Function Documentation

5.15.3.1 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:

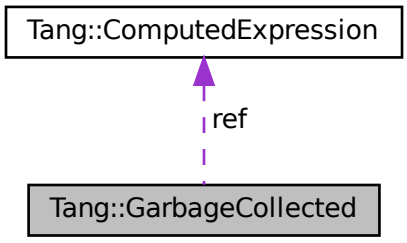
- include/error.hpp
- src/error.cpp

5.16 Tang::GarbageCollected Class Reference

A container that acts as a resource-counting garbage collector for the specified type.

`#include <garbageCollected.hpp>`

Collaboration diagram for Tang::GarbageCollected:



Public Member Functions

- GarbageCollected (const GarbageCollected &other)

  Copy Constructor.
- GarbageCollected (GarbageCollected &&other)

  Move Constructor.
- GarbageCollected & operator= (const GarbageCollected &other)

  Copy Assignment.
- GarbageCollected & operator= (GarbageCollected &&other)

  Move Assignment.
- ~GarbageCollected ()

  Destructor.
- ComputedExpression * operator-> () const

  Access the tracked object as a pointer.
- ComputedExpression & operator* () const

  Access the tracked object.
- bool operator== (const int &val) const

  Compare the GarbageCollected tracked object with a supplied value.
- bool operator== (const double &val) const

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

  Compare the GarbageCollected tracked object with a supplied value.
- bool operator== (const Error &val) const

  Compare the GarbageCollected tracked object with a supplied value.
- 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.

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.

5.16.1 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.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 GarbageCollected() [1/3]

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

Copy Constructor.

Parameters

The	other GarbageCollected object to copy.

5.16.2.2 GarbageCollected() [2/3]

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

Move Constructor.

Parameters

The	other GarbageCollected object to move.

5.16.2.3 ~GarbageCollected()

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

Destructor.

Clean up the tracked object, if appropriate.

5.16.2.4 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.

5.16.3 Member Function Documentation

5.16.3.1 make()

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

Creates a garbage-collected object of the specified type.

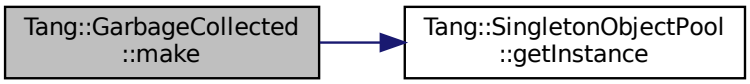
Parameters

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

Returns

A GarbageCollected object.

Here is the call graph for this function:



5.16.3.2 operator%()

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

Perform a modulo between two GarbageCollected values.

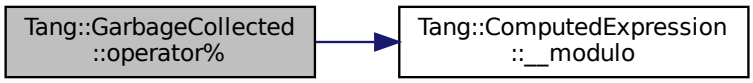
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.16.3.3 operator*() [1/2]

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

Access the tracked object.

Returns

A reference to the tracked object.

5.16.3.4 operator*() [2/2]

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

Perform a multiplication between two GarbageCollected values.

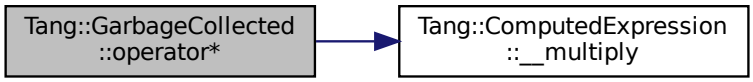
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.16.3.5 operator+()

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

Perform an addition between two GarbageCollected values.

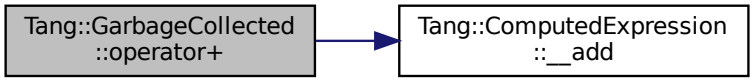
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.16.3.6 operator-() [1/2]

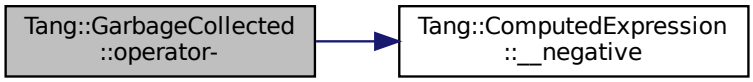
```
GarbageCollected GarbageCollected::operator- ( ) const
```

Perform a negation on the GarbageCollected value.

Returns

The result of the operation.

Here is the call graph for this function:



5.16.3.7 operator-() [2/2]

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

Perform a subtraction between two GarbageCollected values.

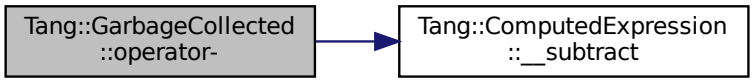
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.16.3.8 operator->()

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

Access the tracked object as a pointer.

Returns

A pointer to the tracked object.

5.16.3.9 operator/()

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

Perform a division between two GarbageCollected values.

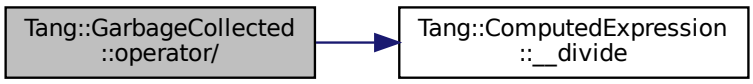
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.16.3.10 operator=() [1/2]

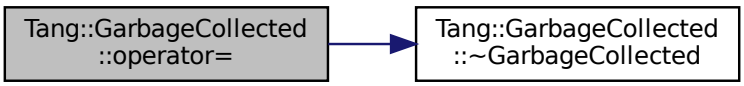
```
GarbageCollected& Tang::GarbageCollected::operator= (
            const GarbageCollected & other )  [inline]
```

Copy Assignment.

Parameters

The	other GarbageCollected object.

Here is the call graph for this function:



5.16.3.11 operator=() [2/2]

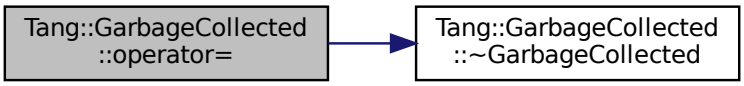
```
GarbageCollected& Tang::GarbageCollected::operator= (
            GarbageCollected && other )  [inline]
```

Move Assignment.

Parameters

The	other GarbageCollected object.

Here is the call graph for this function:



5.16.3.12 operator==() [1/4]

```
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.

5.16.3.13 operator==() [2/4]

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

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.16.3.14 operator==() [3/4]

```
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.

5.16.3.15 operator==() [4/4]

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

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.16.4 Friends And Related Function Documentation

5.16.4.1 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:

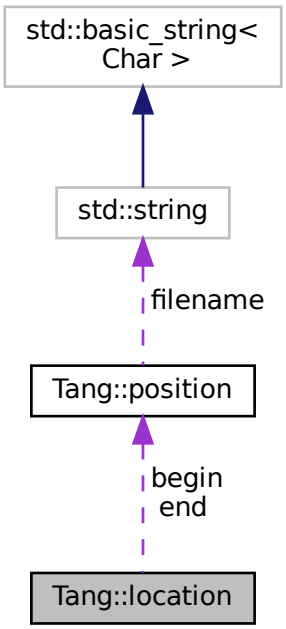
- include/garbageCollected.hpp
- src/garbageCollected.cpp

5.17 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.

5.17.1 Detailed Description

Two points in a source file.

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

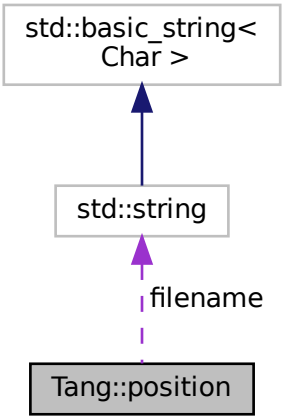
- build/generated/location.hh

5.18 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.

5.18.1 Detailed Description

A point in a source file.

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

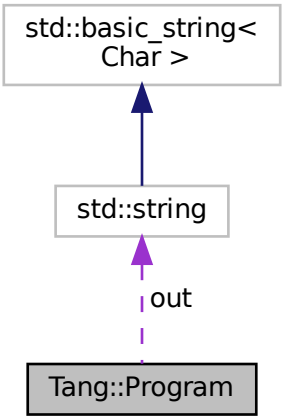
- build/generated/location.hh

5.19 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.
- ~Program ()

  The Program Destructor.
- Program (const Program &program)

  The Copy Constructor.
- Program & operator= (const Program &program)

  The Copy Assignment operator.
- Program (Program &&program)

  The Move Constructor.
- Program & operator= (Program &&program)

  The Move Assignment operator.
- 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.
- void addBytecode (uint64_t)

  Add a uint64_t to the Bytecode.
- Program & execute ()

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

Public Attributes

- std::string out

  The output of the program, resulting from the program execution.

5.19.1 Detailed Description

Represents a compiled script or template that may be executed.

5.19.2 Member Enumeration Documentation

5.19.2.1 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.

5.19.3 Constructor & Destructor Documentation

5.19.3.1 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`.

5.19.4 Member Function Documentation

5.19.4.1 addBytecode()

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

Add a uint64_t to the Bytecode.

Parameters

op	The value to add to the Bytecode.

5.19.4.2 dumpBytecode()

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

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

Returns

A string containing the Opcode representation.

5.19.4.3 execute()

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

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

Returns

The current Program object.

5.19.4.4 getAst()

```
optional< const shared_ptr< AstNode > > Program::getAst ( ) const
```

Get the AST that was generated by the parser.

The parser may have failed, so the return is an `optional<>` type. If the compilation failed, check Program::error.

Returns

A pointer to the AST, if it exists.

5.19.4.5 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.

5.19.4.6 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.

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

5.20 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.

5.20.1 Detailed Description

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

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

5.20.2 Member Function Documentation

5.20.2.1 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.

5.20.2.2 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.

5.20.2.3 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

5.21 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.

5.21.1 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).

5.21.2 Constructor & Destructor Documentation

5.21.2.1 TangBase()

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

The constructor.

Isn't it glorious.

5.21.3 Member Function Documentation

5.21.3.1 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:

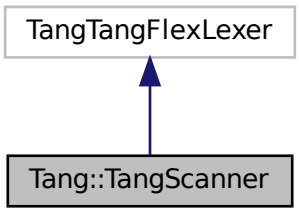
- include/tangBase.hpp
- src/tangBase.cpp

5.22 Tang::TangScanner Class Reference

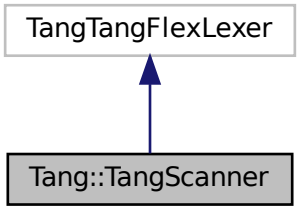
The Flex lexer class for the main Tang language.

`#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.

5.22.1 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.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 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)

5.22.3 Member Function Documentation

5.22.3.1 get_next_token()

```
virtual Tang::TangParser::symbol_type Tang::TangScanner::get_next_token ( )  [virtual]
```

A pass-through function that we supply so that we can provide a Bison 3 token return type instead of the `int` that is returned by the default class configuration.

Returns

A Bison 3 token representing the lexeme that was recognized.

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

- include/tangScanner.hpp

Chapter 6

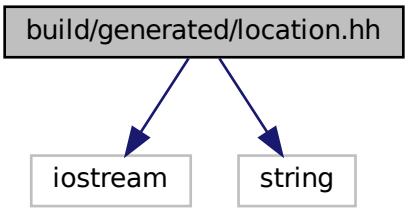
File Documentation

6.1 build/generated/location.hh File Reference

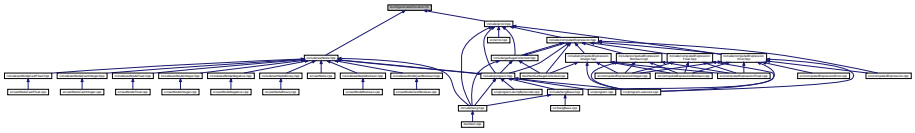
Define the Tang ::location class.

```
#include <iostream>
#include <string>
```

Include dependency graph for location.hh:



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



Classes

- class Tang::position

  A point in a source file.
- class Tang::location

  Two points in a source file.

Macros

- #define **YY_NULLPTR** ((void∗)0)

Functions

- position & Tang::operator+= (position &res, position::counter_type width)

  Add width columns, in place.
- position Tang::operator+ (position res, position::counter_type width)

  Add width columns.
- position & Tang::operator-= (position &res, position::counter_type width)

  Subtract width columns, in place.
- position Tang::operator- (position res, position::counter_type width)

  Subtract width columns.
- template<typename YYChar >
  std::basic_ostream< YYChar > & Tang::operator<< (std::basic_ostream< YYChar > &ostr, const position &pos)

  Intercept output stream redirection.
- location & Tang::operator+= (location &res, const location &end)

  Join two locations, in place.
- location Tang::operator+ (location res, const location &end)

  Join two locations.
- location & Tang::operator+= (location &res, location::counter_type width)

  Add width columns to the end position, in place.
- location Tang::operator+ (location res, location::counter_type width)

  Add width columns to the end position.
- location & Tang::operator-= (location &res, location::counter_type width)

  Subtract width columns to the end position, in place.
- location Tang::operator- (location res, location::counter_type width)

  Subtract width columns to the end position.
- template<typename YYChar >
  std::basic_ostream< YYChar > & Tang::operator<< (std::basic_ostream< YYChar > &ostr, const location &loc)

  Intercept output stream redirection.

6.1.1 Detailed Description

Define the Tang ::location class.

6.1.2 Function Documentation

6.1.2.1 operator<<() [1/2]

```
template<typename YYChar >
std::basic_ostream<YYChar>& Tang::operator<< (
            std::basic_ostream< YYChar > & ostr,
            const location & loc )
```

Intercept output stream redirection.

Parameters

ostr	the destination output stream
loc	a reference to the location to redirect

Avoid duplicate information.

6.1.2.2 operator<<() [2/2]

```
template<typename YYChar >
std::basic_ostream<YYChar>& Tang::operator<< (
            std::basic_ostream< YYChar > & ostr,
            const position & pos )
```

Intercept output stream redirection.

Parameters

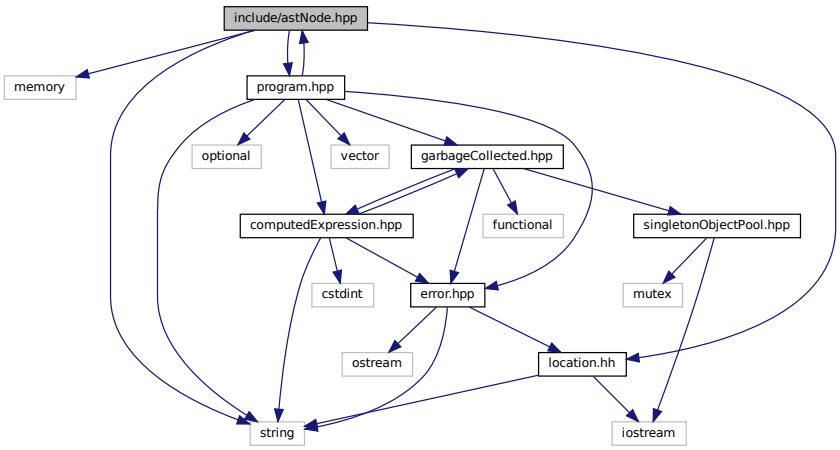
ostr	the destination output stream
pos	a reference to the position to redirect

6.2 include/astNode.hpp File Reference

Declare the Tang::AstNode base class.

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

Include dependency graph for astNode.hpp:



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

Classes

- class Tang::AstNode

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

6.2.1 Detailed Description

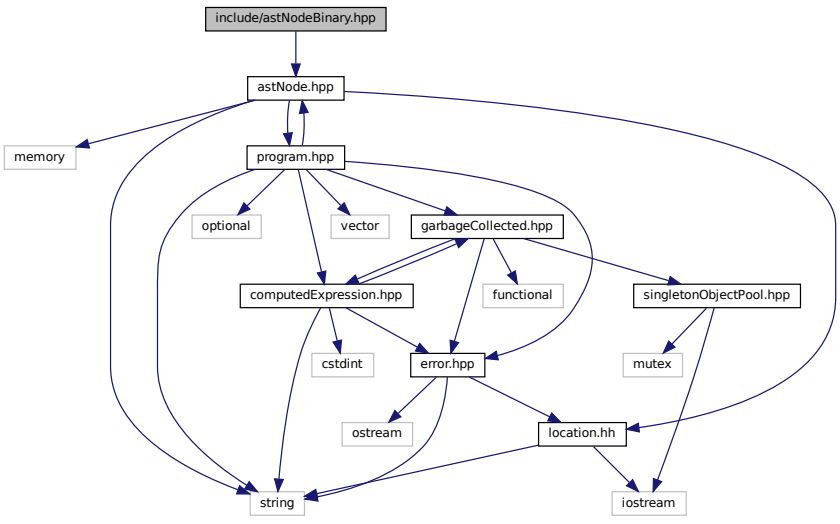
Declare the Tang::AstNode base class.

6.3 include/astNodeBinary.hpp File Reference

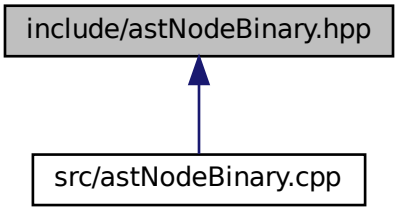
Declare the Tang::AstNodeBinary class.

```
#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.

6.3.1 Detailed Description

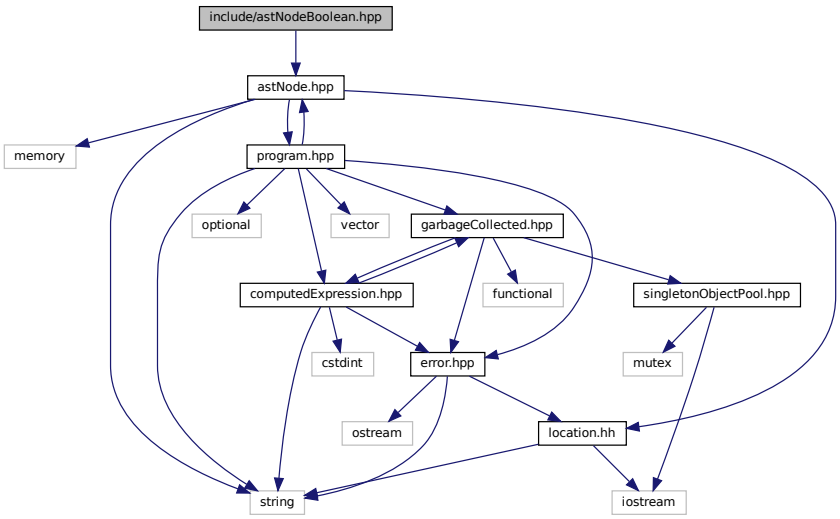
Declare the Tang::AstNodeBinary class.

6.4 include/astNodeBoolean.hpp File Reference

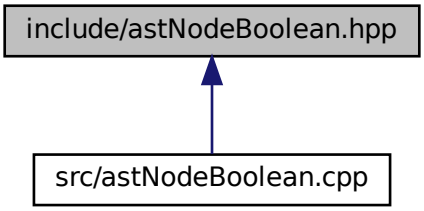
Declare the Tang::AstNodeBoolean class.

```
#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.

6.4.1 Detailed Description

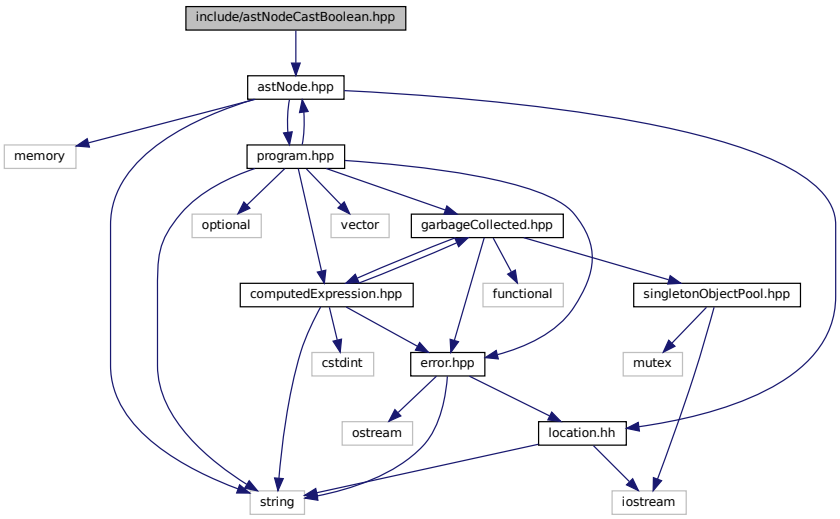
Declare the Tang::AstNodeBoolean class.

6.5 include/astNodeCastBoolean.hpp File Reference

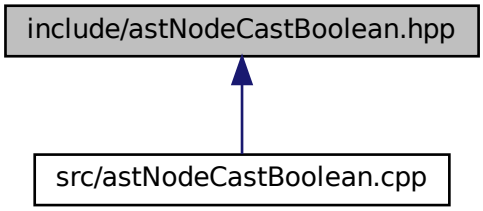
Declare the Tang::AstNodeBoolean class.

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

Include dependency graph for astNodeCastBoolean.hpp:



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



Classes

- class Tang::AstNodeCastBoolean

  An AstNode that represents a typecast to a boolean.

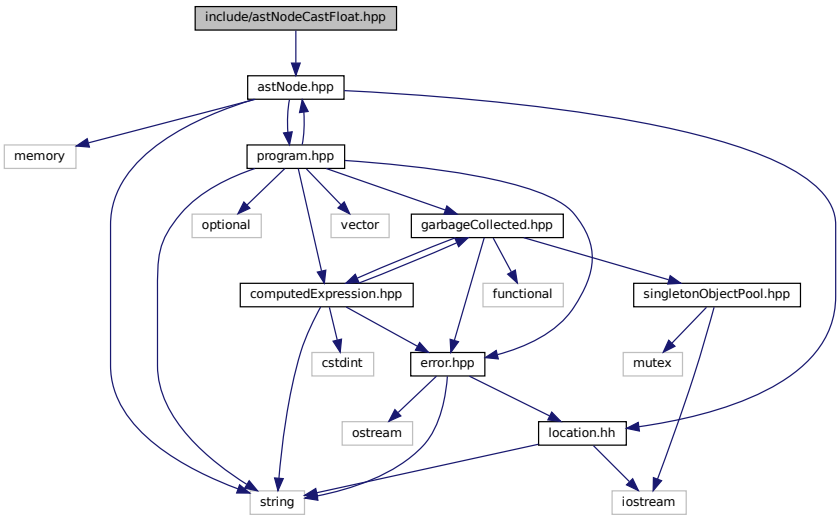
6.5.1 Detailed Description

Declare the Tang::AstNodeBoolean class.

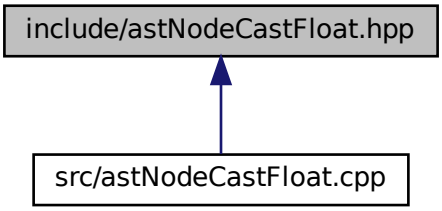
6.6 include/astNodeCastFloat.hpp File Reference

Declare the Tang::AstNodeFloat class.

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



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



Classes

- class Tang::AstNodeCastFloat

  An AstNode that represents a typecast to a float.

6.6.1 Detailed Description

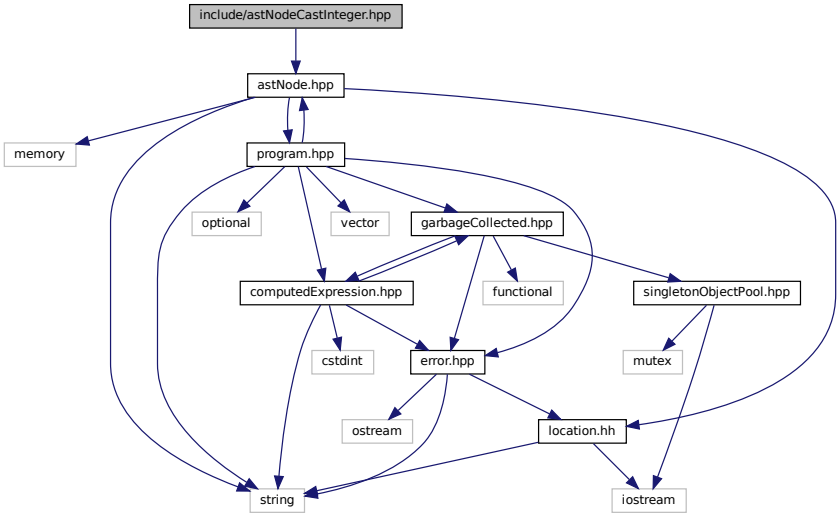
Declare the Tang::AstNodeFloat class.

6.7 include/astNodeCastInteger.hpp File Reference

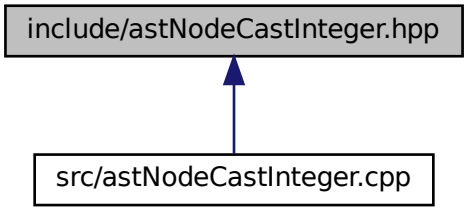
Declare the Tang::AstNodeInteger class.

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

Include dependency graph for astNodeCastInteger.hpp:



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



Classes

- class Tang::AstNodeCastInteger

  An AstNode that represents a typecast to an integer.

6.7.1 Detailed Description

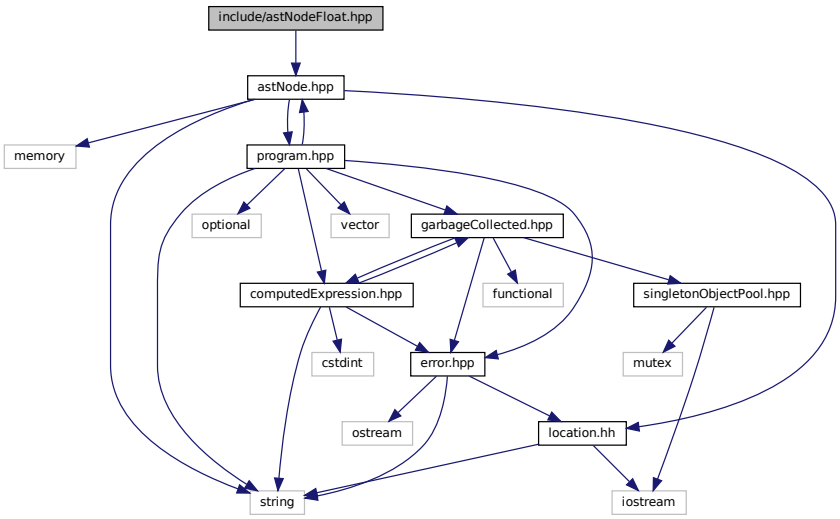
Declare the Tang::AstNodeInteger class.

6.8 include/astNodeFloat.hpp File Reference

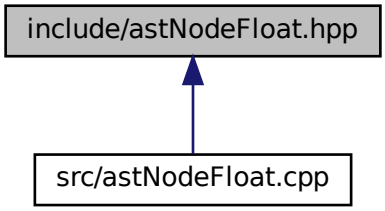
Declare the Tang::AstNodeFloat class.

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

Include dependency graph for astNodeFloat.hpp:



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



Classes

- class Tang::AstNodeFloat

  An AstNode that represents an float literal.

6.8.1 Detailed Description

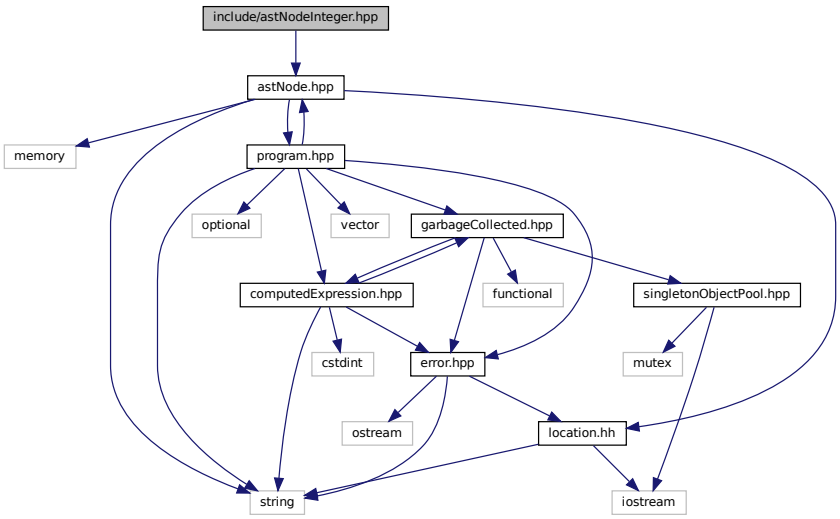
Declare the Tang::AstNodeFloat class.

6.9 include/astNodeInteger.hpp File Reference

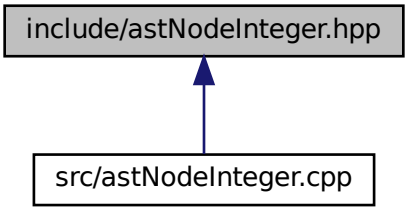
Declare the Tang::AstNodeInteger class.

```
#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.

6.9.1 Detailed Description

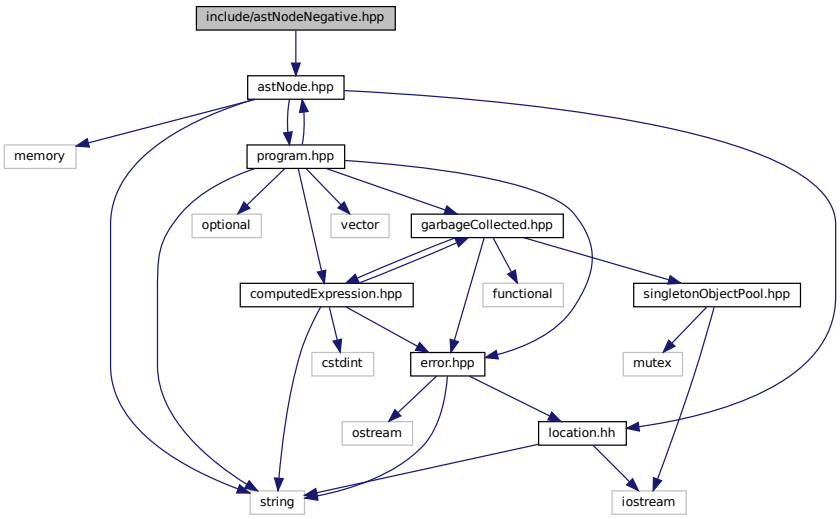
Declare the Tang::AstNodeInteger class.

6.10 include/astNodeNegative.hpp File Reference

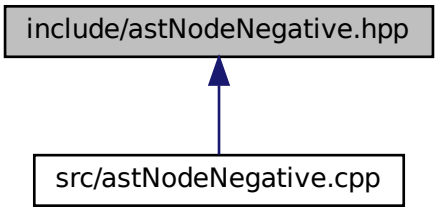
Declare the Tang::AstNodeNegative class.

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

Include dependency graph for astNodeNegative.hpp:



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



Classes

- class Tang::AstNodeNegative

  An AstNode that represents a unary negation.

6.10.1 Detailed Description

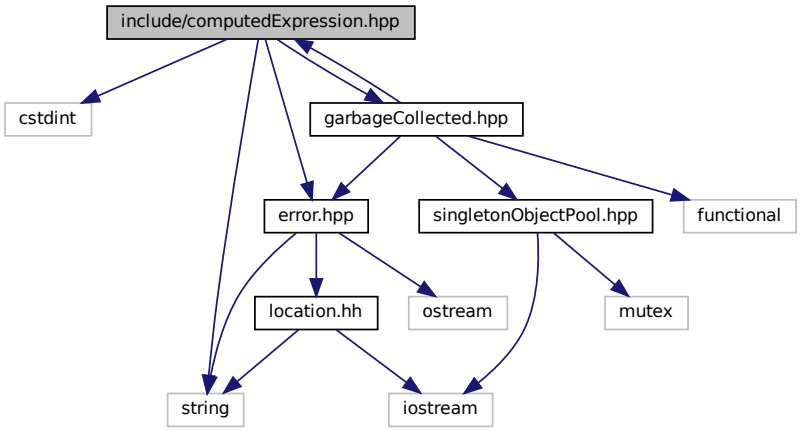
Declare the Tang::AstNodeNegative class.

6.11 include/computedExpression.hpp File Reference

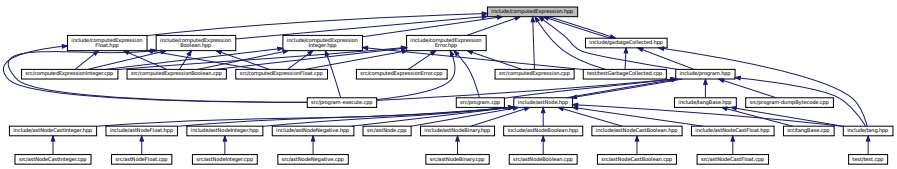
Declare the Tang::ComputedExpression base class.

```
#include <cstdint>
#include <string>
#include "garbageCollected.hpp"
#include "error.hpp"
```

Include dependency graph for computedExpression.hpp:



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



Classes

- class Tang::ComputedExpression

  Represents the result of a computation that has been executed.

6.11.1 Detailed Description

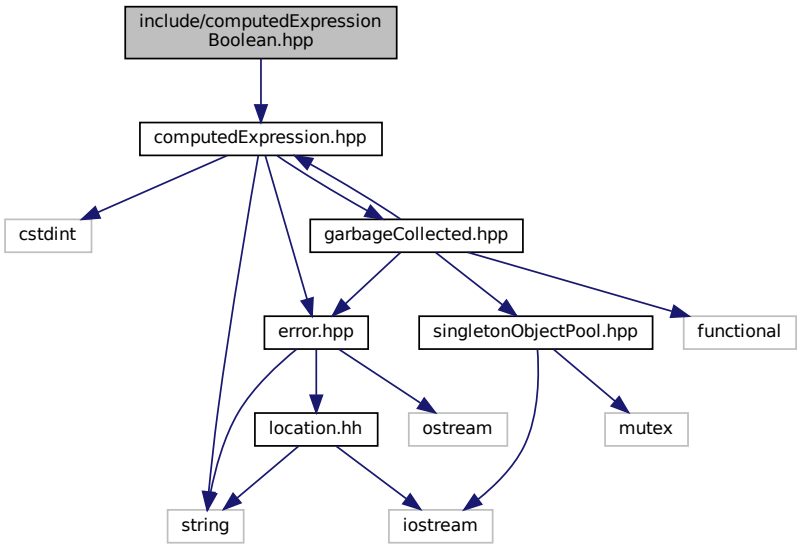
Declare the Tang::ComputedExpression base class.

6.12 include/computedExpressionBoolean.hpp File Reference

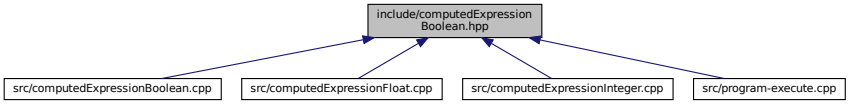
Declare the Tang::ComputedExpressionBoolean class.

```
#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.

6.12.1 Detailed Description

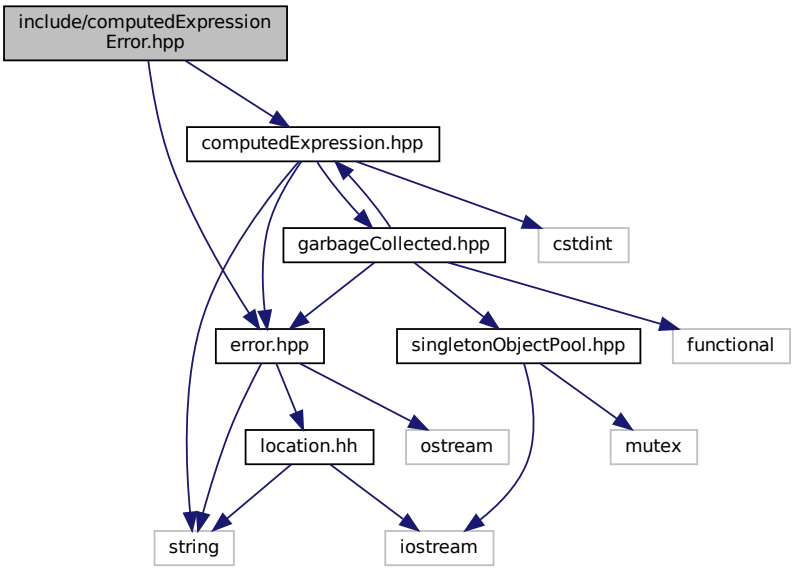
Declare the Tang::ComputedExpressionBoolean class.

6.13 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.

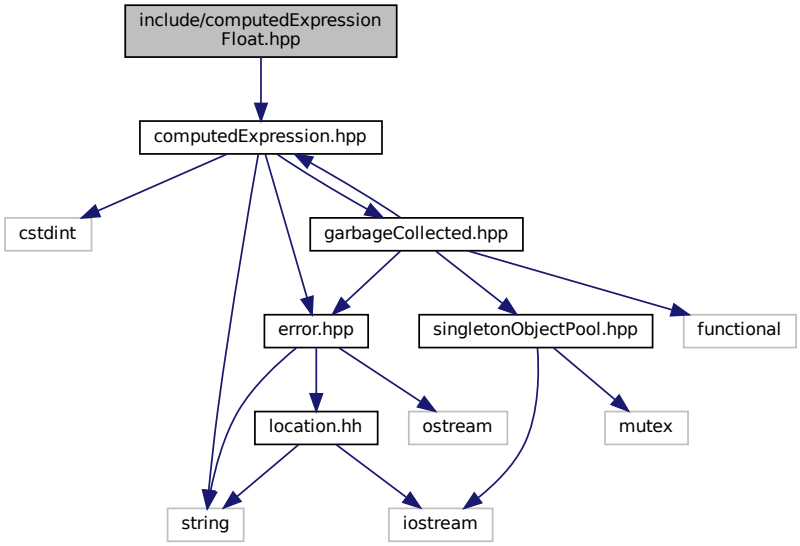
6.13.1 Detailed Description

Declare the Tang::ComputedExpressionError class.

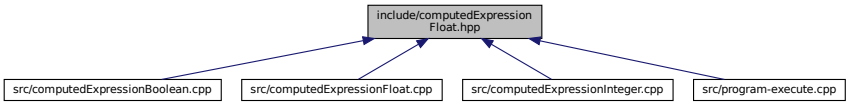
6.14 include/computedExpressionFloat.hpp File Reference

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.

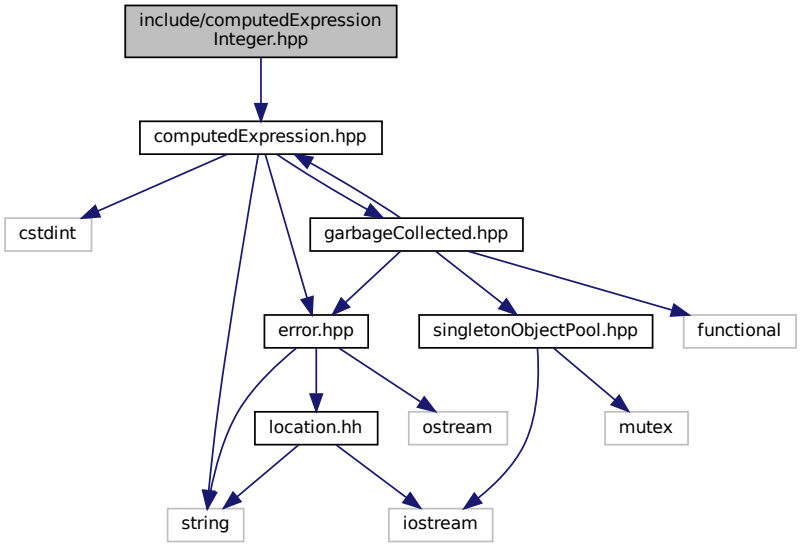
6.14.1 Detailed Description

Declare the Tang::ComputedExpressionFloat class.

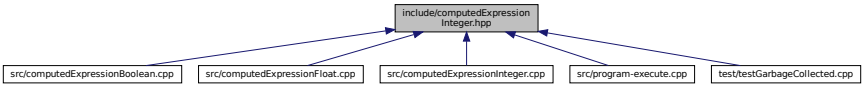
6.15 include/computedExpressionInteger.hpp File Reference

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.

6.15.1 Detailed Description

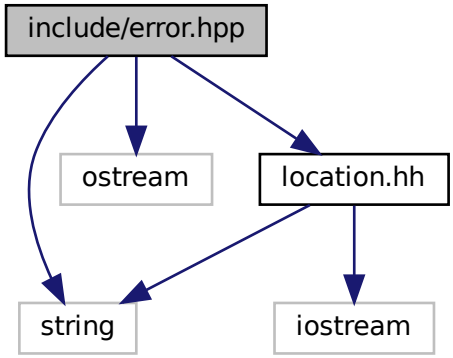
Declare the Tang::ComputedExpressionInteger class.

6.16 include/error.hpp File Reference

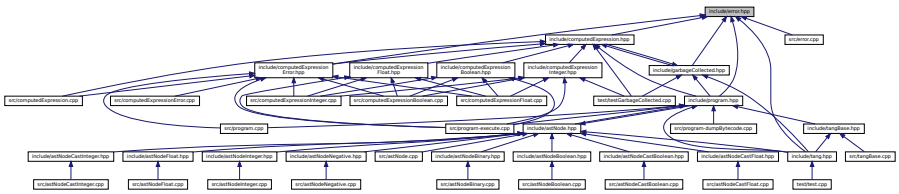
Declare the Tang::Error class used to describe syntax and runtime errors.

```
#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.

6.16.1 Detailed Description

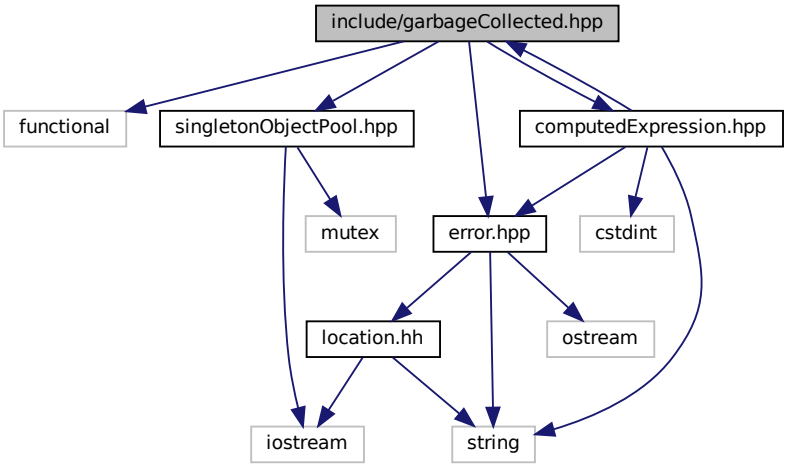
Declare the Tang::Error class used to describe syntax and runtime errors.

6.17 include/garbageCollected.hpp File Reference

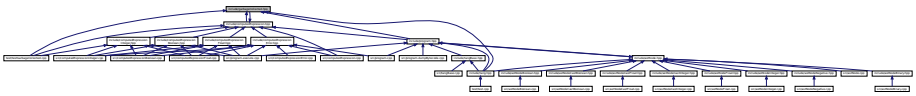
Declare the Tang::GarbageCollected class.

```
#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.

6.17.1 Detailed Description

Declare the Tang::GarbageCollected class.

6.18 include/macros.hpp File Reference

Contains generic macros.

Macros

- #define TANG_UNUSED(x) x

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

6.18.1 Detailed Description

Contains generic macros.

6.18.2 Macro Definition Documentation

6.18.2.1 TANG_UNUSED

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

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

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

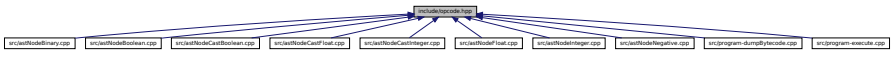
Parameters

x	The argument to be ignored.

6.19 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 {
  INTEGER , FLOAT , BOOLEAN , ADD ,
  SUBTRACT , MULTIPLY , DIVIDE , MODULO ,
  NEGATIVE , CASTINTEGER , CASTFLOAT , CASTBOOLEAN }

6.19.1 Detailed Description

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

6.19.2 Enumeration Type Documentation

6.19.2.1 Opcode

```
enum Tang::Opcode  [strong]
```

Enumerator

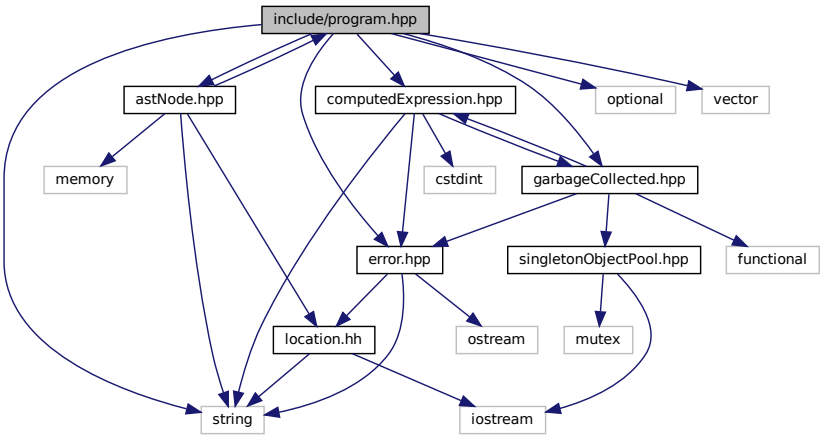
INTEGER	Push an integer onto the stack.
FLOAT	Push a floating point number onto the stack.
BOOLEAN	Push a boolean onto the stack.
ADD	Pop rhs, pop lhs, push lhs + rhs.
SUBTRACT	Pop rhs, pop lhs, push lhs - rhs.
MULTIPLY	Pop rhs, pop lhs, push lhs ∗ rhs.
DIVIDE	Pop rhs, pop lhs, push lhs / rhs.
MODULO	Pop rhs, pop lhs, push lhs % rhs.
NEGATIVE	Pop val, push negative val.
CASTINTEGER	Pop a val, typecast to int, push.
CASTFLOAT	Pop a val, typecast to float, push.
CASTBOOLEAN	Pop a val, typecast to boolean, push.

6.20 include/program.hpp File Reference

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

```
#include <string>
#include <optional>
#include <vector>
#include "astNode.hpp"
#include "error.hpp"
#include "computedExpression.hpp"
#include "garbageCollected.hpp"
```

Include dependency graph for program.hpp:



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

Classes

- class Tang::Program

  Represents a compiled script or template that may be executed.

Typedefs

- using Tang::Bytecode = std::vector< uint64_t >

  Contains the Opcodes of a compiled program.

6.20.1 Detailed Description

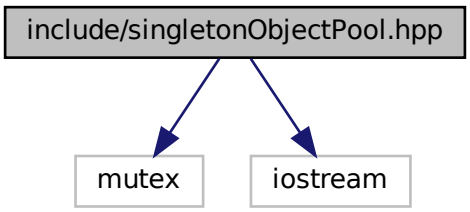
Declare the Tang::Program class used to compile and execute source code.

6.21 include/singletonObjectPool.hpp File Reference

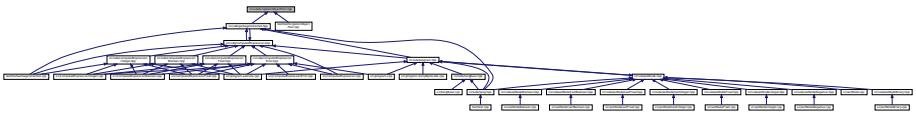
Declare the Tang::SingletonObjectPool class.

```
#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.

6.21.1 Detailed Description

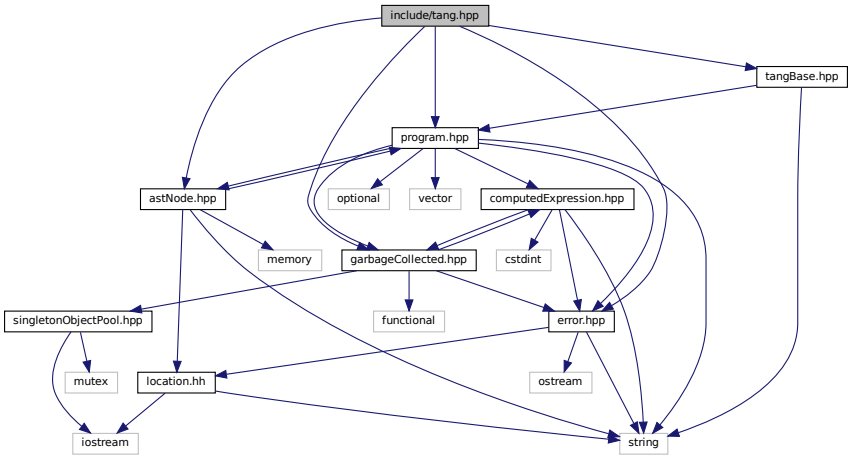
Declare the Tang::SingletonObjectPool class.

6.22 include/tang.hpp File Reference

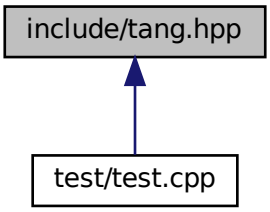
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

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

Include dependency graph for tang.hpp:



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



6.22.1 Detailed Description

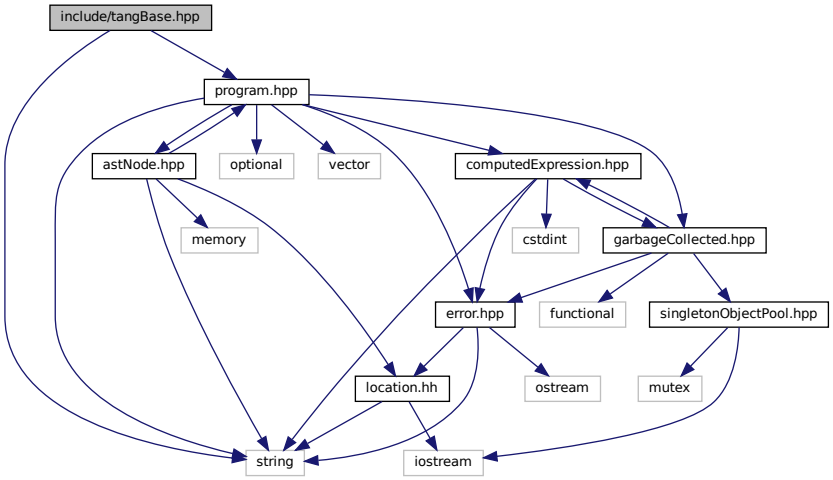
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

6.23 include/tangBase.hpp File Reference

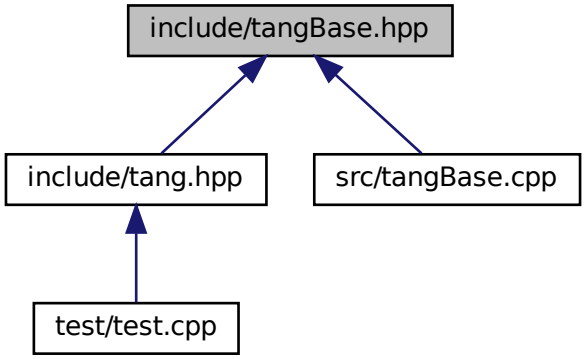
Declare the Tang::TangBase class used to interact with Tang.

```
#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.

6.23.1 Detailed Description

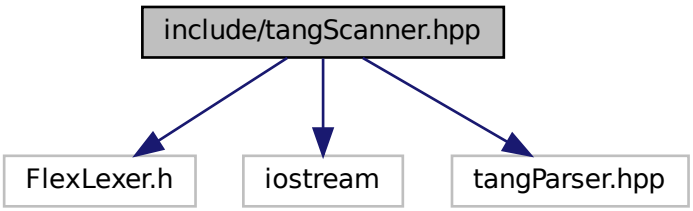
Declare the Tang::TangBase class used to interact with Tang.

6.24 include/tangScanner.hpp File Reference

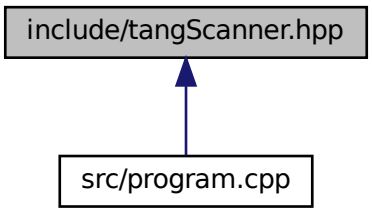
Declare the Tang::TangScanner used to tokenize a Tang script.

```
#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()

6.24.1 Detailed Description

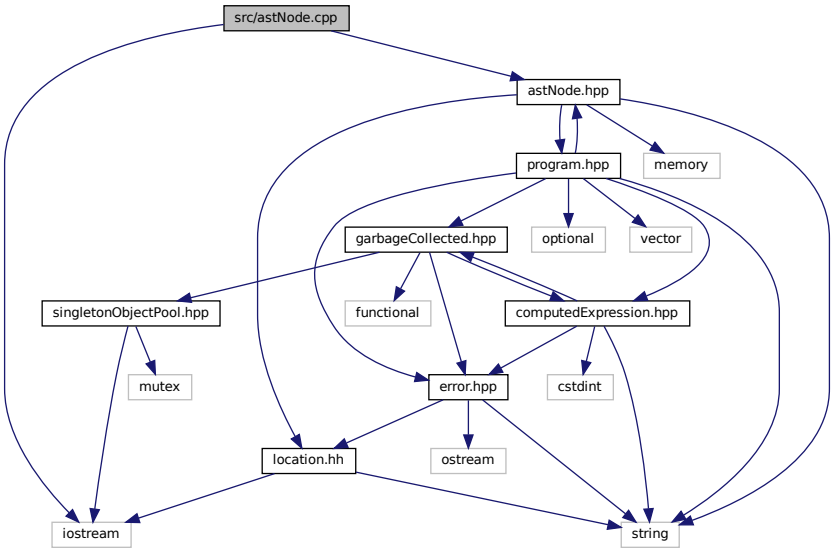
Declare the Tang::TangScanner used to tokenize a Tang script.

6.25 src/astNode.cpp File Reference

Define the Tang::AstNode class.

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

Include dependency graph for astNode.cpp:



6.25.1 Detailed Description

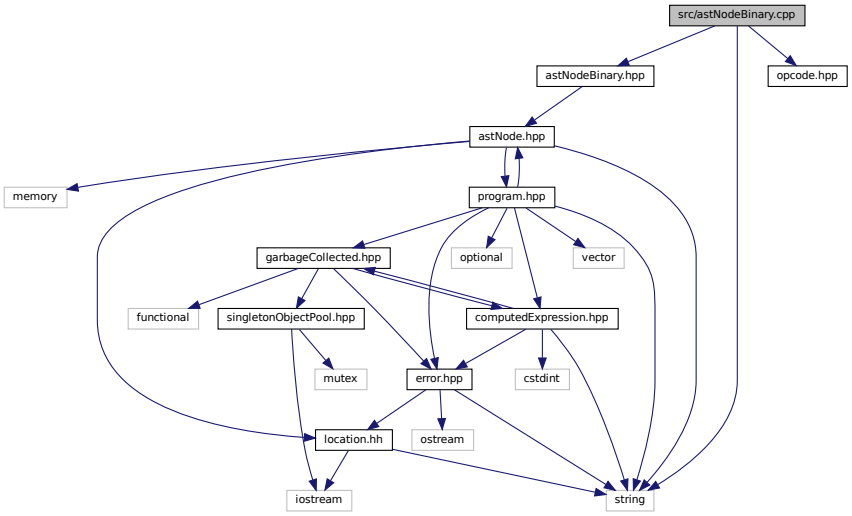
Define the Tang::AstNode class.

6.26 src/astNodeBinary.cpp File Reference

Define the Tang::AstNodeBinary class.

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

Include dependency graph for astNodeBinary.cpp:



6.26.1 Detailed Description

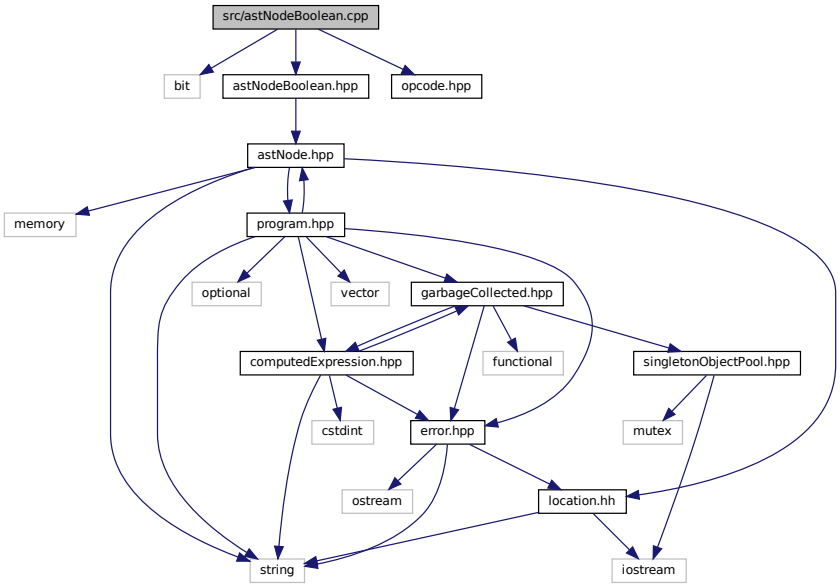
Define the Tang::AstNodeBinary class.

6.27 src/astNodeBoolean.cpp File Reference

Define the Tang::AstNodeBoolean class.

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

Include dependency graph for astNodeBoolean.cpp:



6.27.1 Detailed Description

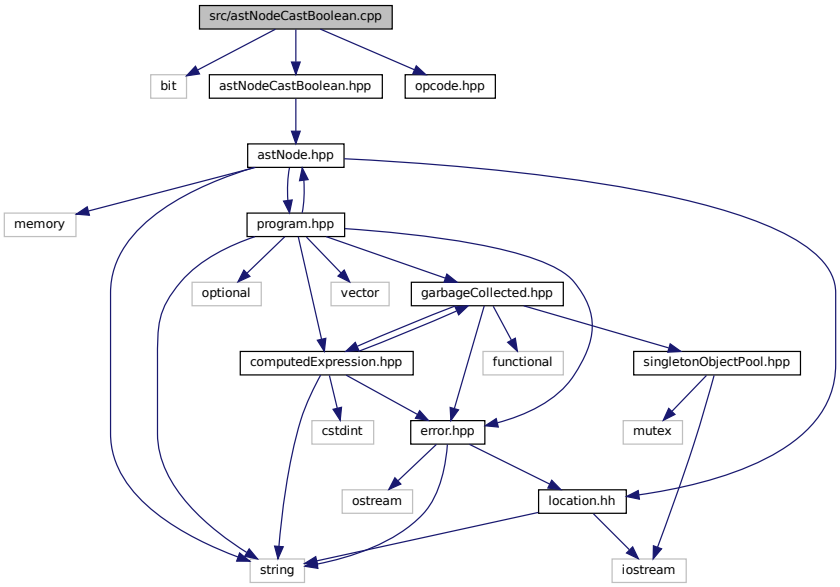
Define the Tang::AstNodeBoolean class.

6.28 src/astNodeCastBoolean.cpp File Reference

Define the Tang::AstNodeCastBoolean class.

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

Include dependency graph for astNodeCastBoolean.cpp:



6.28.1 Detailed Description

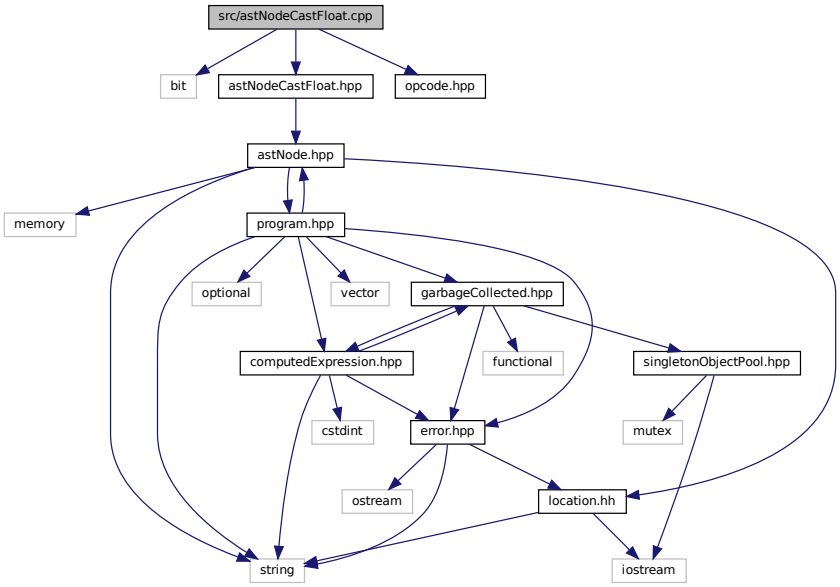
Define the Tang::AstNodeCastBoolean class.

6.29 src/astNodeCastFloat.cpp File Reference

Define the Tang::shared_ptr<AstNode>CastFloat class.

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

Include dependency graph for astNodeCastFloat.cpp:



6.29.1 Detailed Description

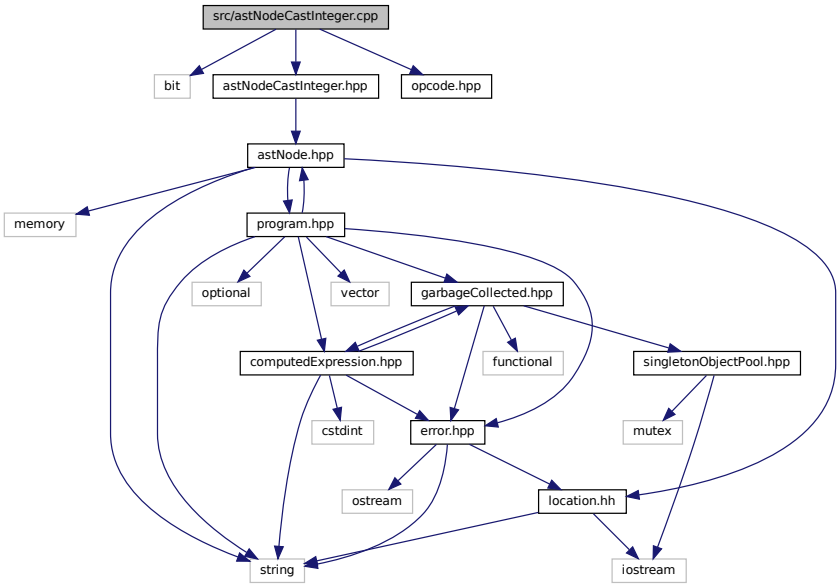
Define the Tang::shared_ptr<AstNode>CastFloat class.

6.30 src/astNodeCastInteger.cpp File Reference

Define the Tang::AstNodeCastInteger class.

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

Include dependency graph for astNodeCastInteger.cpp:



6.30.1 Detailed Description

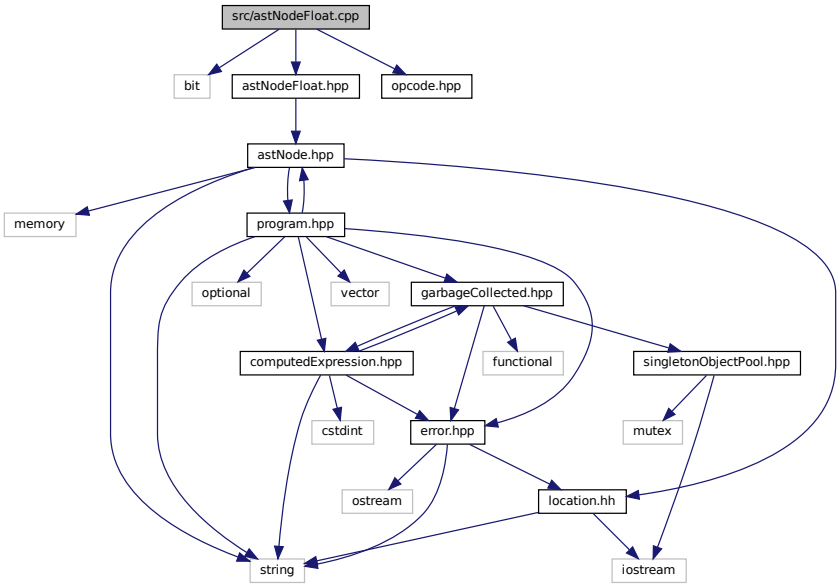
Define the Tang::AstNodeCastInteger class.

6.31 src/astNodeFloat.cpp File Reference

Define the Tang::AstNodeFloat class.

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

Include dependency graph for astNodeFloat.cpp:



6.31.1 Detailed Description

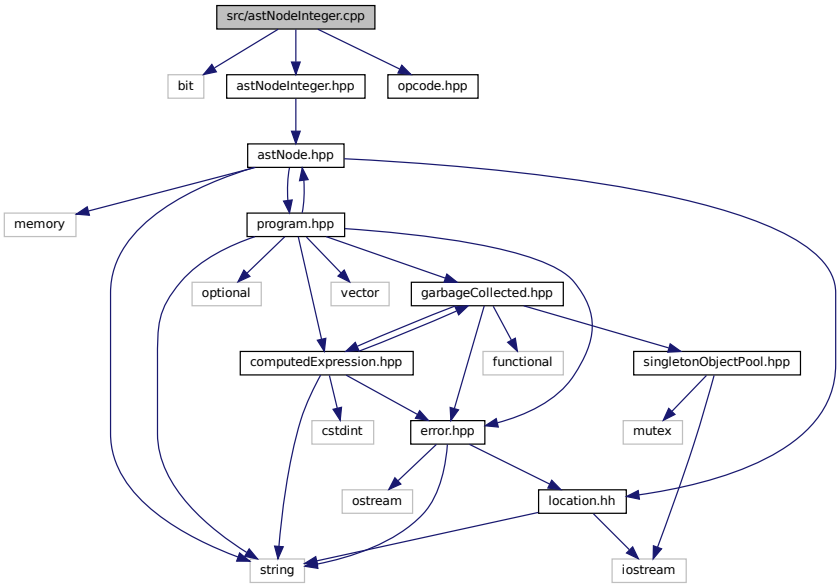
Define the Tang::AstNodeFloat class.

6.32 src/astNodeInteger.cpp File Reference

Define the Tang::AstNodeInteger class.

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

Include dependency graph for astNodeInteger.cpp:



6.32.1 Detailed Description

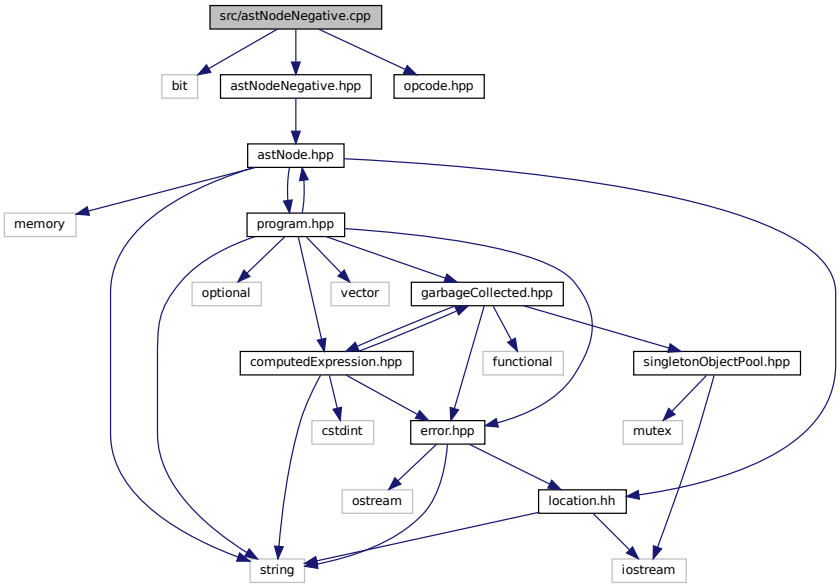
Define the Tang::AstNodeInteger class.

6.33 src/astNodeNegative.cpp File Reference

Define the Tang::AstNodeNegative class.

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

Include dependency graph for astNodeNegative.cpp:



6.33.1 Detailed Description

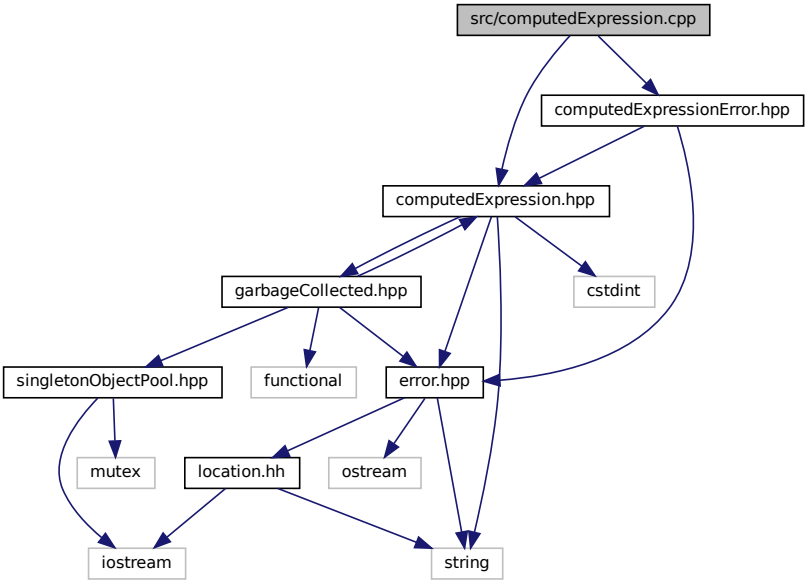
Define the Tang::AstNodeNegative class.

6.34 src/computedExpression.cpp File Reference

Define the Tang::ComputedExpression class.

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

Include dependency graph for computedExpression.cpp:



6.34.1 Detailed Description

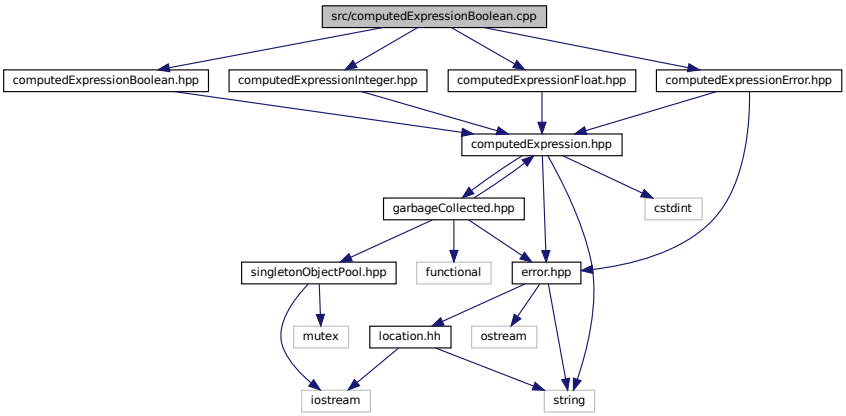
Define the Tang::ComputedExpression class.

6.35 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:



6.35.1 Detailed Description

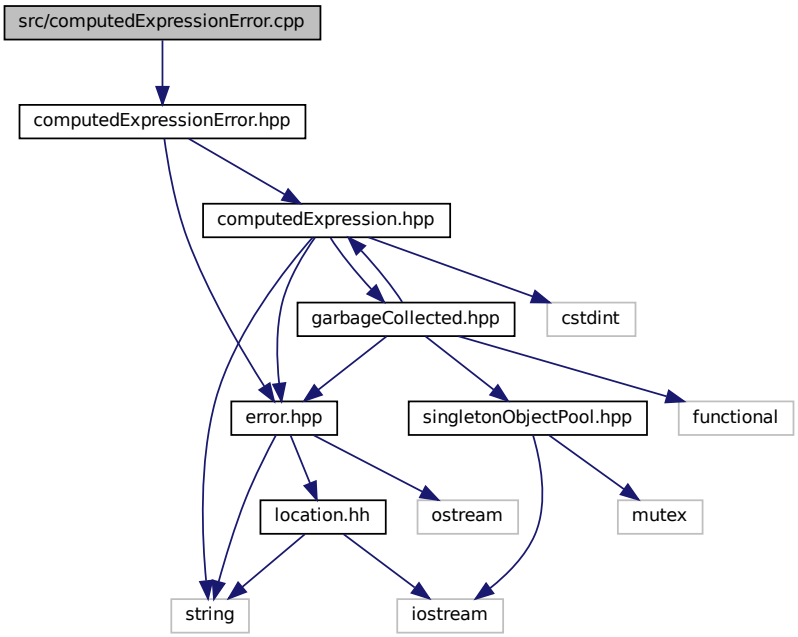
Define the Tang::ComputedExpressionBoolean class.

6.36 src/computedExpressionError.cpp File Reference

Define the Tang::ComputedExpressionError class.

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

Include dependency graph for computedExpressionError.cpp:



6.36.1 Detailed Description

Define the Tang::ComputedExpressionError class.

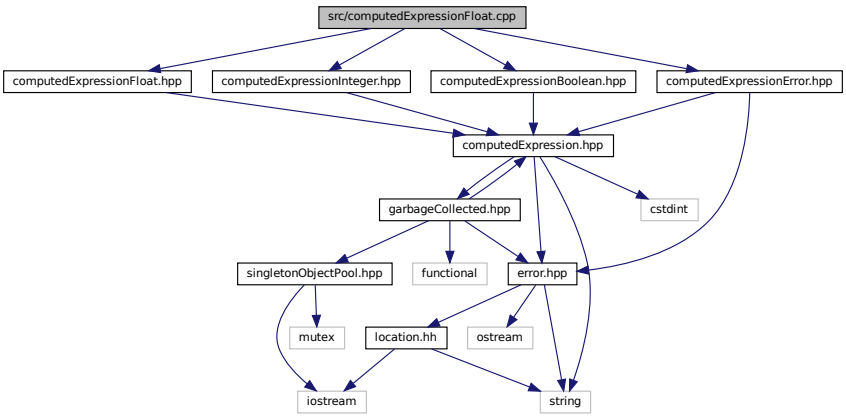
6.37 src/computedExpressionFloat.cpp File Reference

Define the Tang::ComputedExpressionFloat class.

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

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

Include dependency graph for computedExpressionFloat.cpp:



6.37.1 Detailed Description

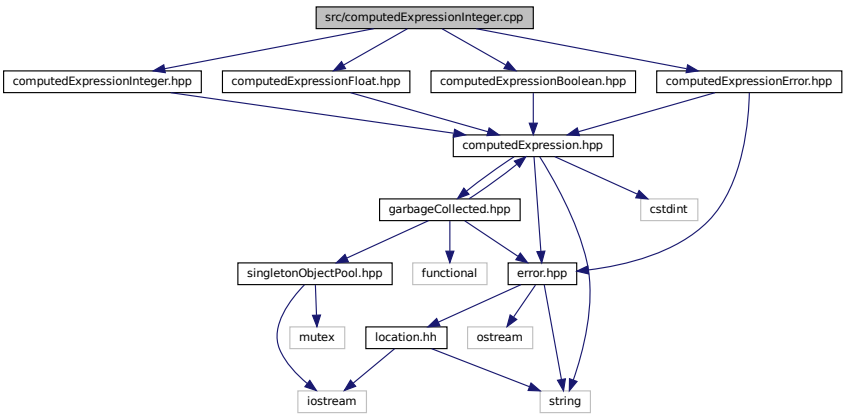
Define the Tang::ComputedExpressionFloat class.

6.38 src/computedExpressionInteger.cpp File Reference

Define the Tang::ComputedExpressionInteger class.

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

Include dependency graph for computedExpressionInteger.cpp:



6.38.1 Detailed Description

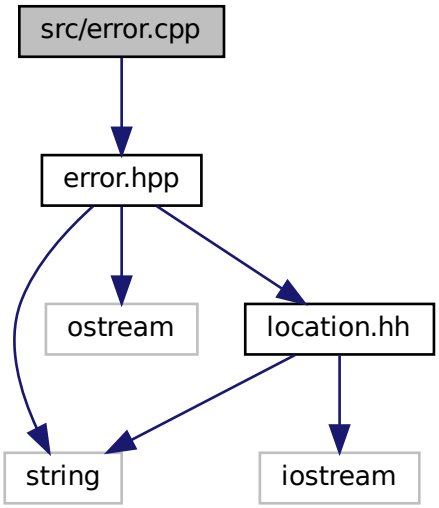
Define the Tang::ComputedExpressionInteger class.

6.39 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)

6.39.1 Detailed Description

Define the Tang::Error class.

6.39.2 Function Documentation

6.39.2.1 operator<<()

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

Parameters

out	The output stream.
error	The Error object.

Returns

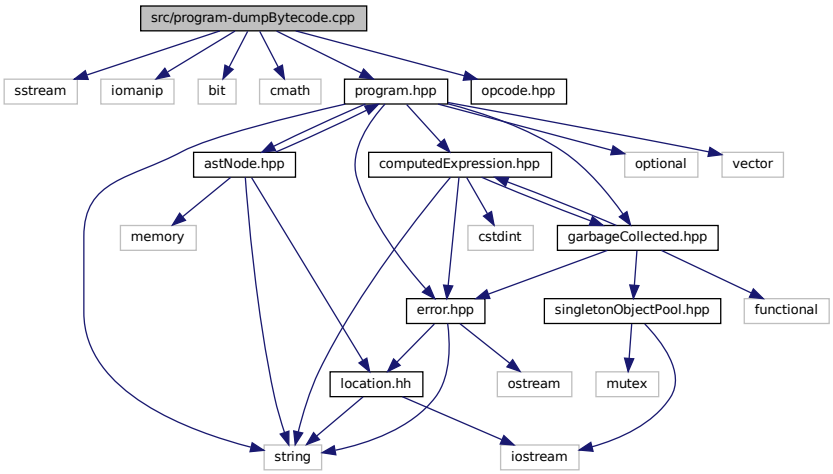
The output stream.

6.40 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.

6.40.1 Detailed Description

Define the Tang::Program::dumpBytecode method.

6.40.2 Macro Definition Documentation

6.40.2.1 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

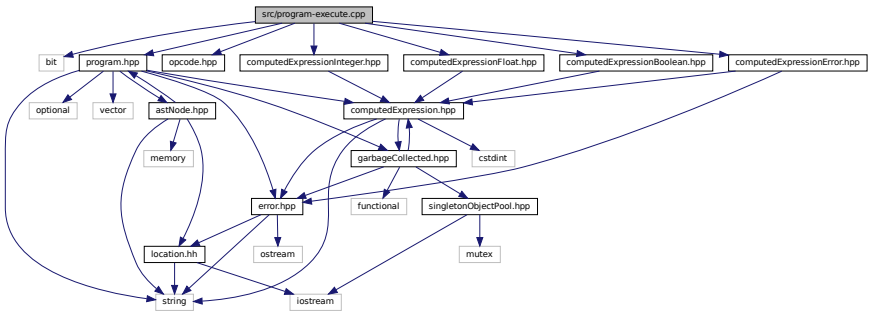
x	The number of additional vector entries that should exist.

6.41 src/program-execute.cpp File Reference

Define the Tang::Program::execute method.

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

Include dependency graph for program-execute.cpp:



Macros

- #define EXECUTEPROGRAMCHECK(x)

Verify the size of the Bytecode vector so that it may be safely accessed.

- #define STACKCHECK(x)

  Verify the size of the stack vector so that it may be safely accessed.

6.41.1 Detailed Description

Define the Tang::Program::execute method.

6.41.2 Macro Definition Documentation

6.41.2.1 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.

6.41.2.2 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

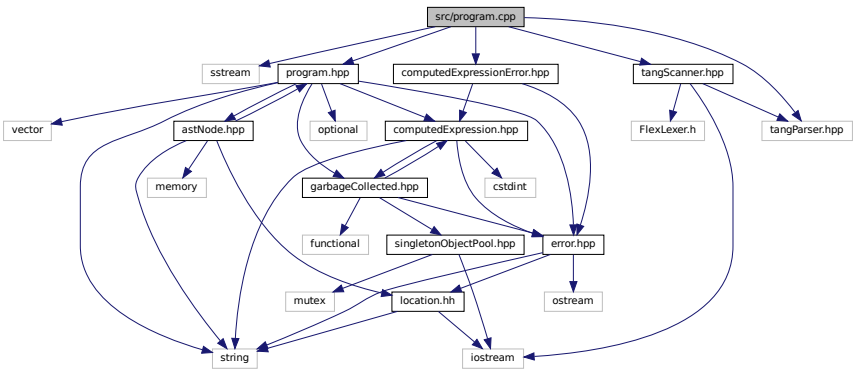
x	The number of entries that should exist in the stack.

6.42 src/program.cpp File Reference

Define the Tang::Program class.

```
#include <sstream>
#include "program.hpp"
#include "tangScanner.hpp"
#include "tangParser.hpp"
#include "computedExpressionError.hpp"
```

Include dependency graph for program.cpp:



6.42.1 Detailed Description

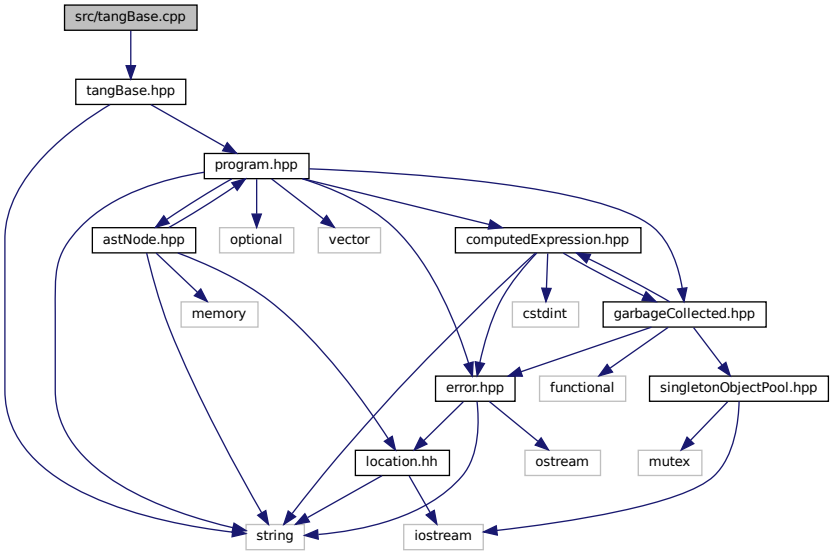
Define the Tang::Program class.

6.43 src/tangBase.cpp File Reference

Define the Tang::TangBase class.

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

Include dependency graph for tangBase.cpp:



6.43.1 Detailed Description

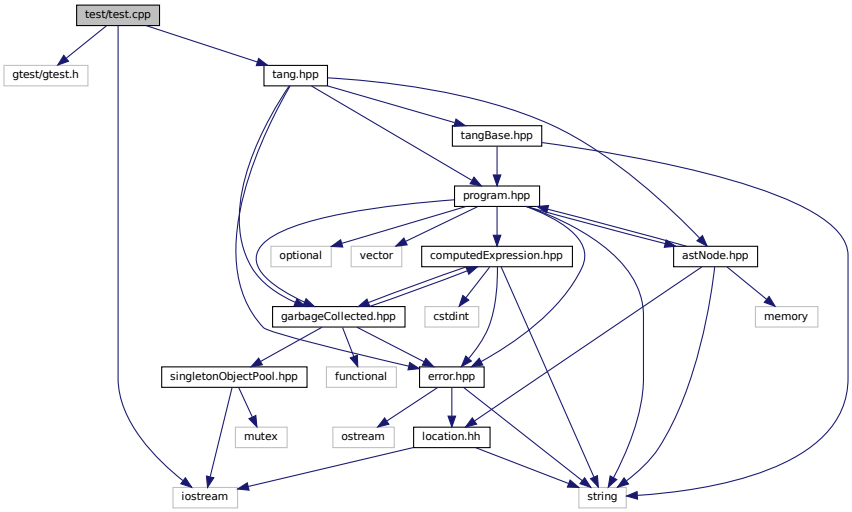
Define the Tang::TangBase class.

6.44 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, Integer)
- **TEST** (Declare, Float)
- **TEST** (Expression, Add)
- **TEST** (Expression, Subtract)
- **TEST** (Expression, Multiplication)
- **TEST** (Expression, Division)
- **TEST** (Expression, Modulo)
- **TEST** (Expression, UnaryMinus)
- **TEST** (Expression, Parentheses)
- **TEST** (Expression, TypeCast)
- **TEST** (Expression, Boolean)
- int **main** (int argc, char ∗∗argv)

6.44.1 Detailed Description

Test the general language behaviors.

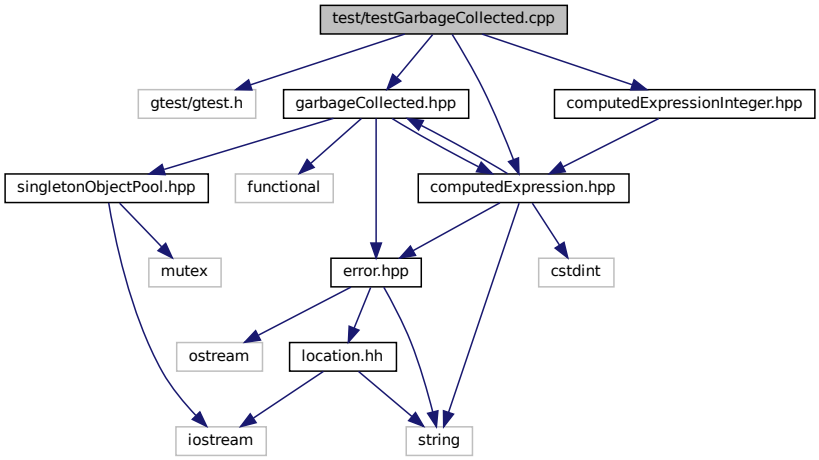
6.45 test/testGarbageCollected.cpp File Reference

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)

6.45.1 Detailed Description

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

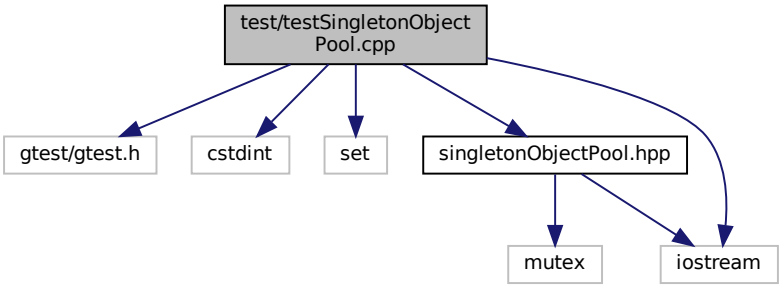
6.46 test/testSingletonObjectPool.cpp File Reference

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)

6.46.1 Detailed Description

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

Index

__add
Tang::ComputedExpression, 37
Tang::ComputedExpressionBoolean, 45
Tang::ComputedExpressionError, 52
Tang::ComputedExpressionFloat, 60
Tang::ComputedExpressionInteger, 68
__boolean
Tang::ComputedExpression, 37
Tang::ComputedExpressionBoolean, 45
Tang::ComputedExpressionError, 52
Tang::ComputedExpressionFloat, 60
Tang::ComputedExpressionInteger, 68
__divide
Tang::ComputedExpression, 37
Tang::ComputedExpressionBoolean, 45
Tang::ComputedExpressionError, 53
Tang::ComputedExpressionFloat, 61
Tang::ComputedExpressionInteger, 68
__float
Tang::ComputedExpression, 38
Tang::ComputedExpressionBoolean, 46
Tang::ComputedExpressionError, 53
Tang::ComputedExpressionFloat, 61
Tang::ComputedExpressionInteger, 69
__integer
Tang::ComputedExpression, 38
Tang::ComputedExpressionBoolean, 46
Tang::ComputedExpressionError, 53
Tang::ComputedExpressionFloat, 61
Tang::ComputedExpressionInteger, 69
__modulo
Tang::ComputedExpression, 38
Tang::ComputedExpressionBoolean, 46
Tang::ComputedExpressionError, 53
Tang::ComputedExpressionFloat, 62
Tang::ComputedExpressionInteger, 69
__multiply
Tang::ComputedExpression, 39
Tang::ComputedExpressionBoolean, 47
Tang::ComputedExpressionError, 55
Tang::ComputedExpressionFloat, 62
Tang::ComputedExpressionInteger, 70
__negative
Tang::ComputedExpression, 39
Tang::ComputedExpressionBoolean, 47
Tang::ComputedExpressionError, 55
Tang::ComputedExpressionFloat, 62
Tang::ComputedExpressionInteger, 70
__subtract
Tang::ComputedExpression, 39
Tang::ComputedExpressionBoolean, 47
Tang::ComputedExpressionError, 55
Tang::ComputedExpressionFloat, 63
Tang::ComputedExpressionInteger, 70
~GarbageCollected
Tang::GarbageCollected, 79

ADD
opcode.hpp, 118
addBytecode
Tang::Program, 92
AstNode
Tang::AstNode, 11
AstNodeBinary
Tang::AstNodeBinary, 14
AstNodeBoolean
Tang::AstNodeBoolean, 17
AstNodeCastBoolean
Tang::AstNodeCastBoolean, 20
AstNodeCastFloat
Tang::AstNodeCastFloat, 23
AstNodeCastInteger
Tang::AstNodeCastInteger, 26
AstNodeFloat
Tang::AstNodeFloat, 29
AstNodeInteger
Tang::AstNodeInteger, 32
AstNodeNegative
Tang::AstNodeNegative, 35

BOOLEAN
opcode.hpp, 118
build/generated/location.hh, 99

CASTBOOLEAN
opcode.hpp, 118
CASTFLOAT
opcode.hpp, 118
CASTINTEGER
opcode.hpp, 118
CodeType
Tang::Program, 91
compileScript
Tang::TangBase, 96
ComputedExpressionBoolean
Tang::ComputedExpressionBoolean, 44
ComputedExpressionError
Tang::ComputedExpressionError, 52
ComputedExpressionFloat

Tang::ComputedExpressionFloat, 60
ComputedExpressionInteger
Tang::ComputedExpressionInteger, 67

DIVIDE
opcode.hpp, 118
dump
Tang::ComputedExpression, 40
Tang::ComputedExpressionBoolean, 48
Tang::ComputedExpressionError, 56
Tang::ComputedExpressionFloat, 63
Tang::ComputedExpressionInteger, 71
dumpBytecode
Tang::Program, 92
DUMPPROGRAMCHECK
program-dumpBytecode.cpp, 138

Error
Tang::Error, 75
error.cpp
operator<<, 136
execute
Tang::Program, 93
EXECUTEPROGRAMCHECK
program-execute.cpp, 139

FLOAT
opcode.hpp, 118

GarbageCollected
Tang::GarbageCollected, 78, 79
get
Tang::SingletonObjectPool< T >, 94
get_next_token
Tang::TangScanner, 98
getAst
Tang::Program, 93
getCode
Tang::Program, 93
getInstance
Tang::SingletonObjectPool< T >, 95
getResult
Tang::Program, 93

include/astNode.hpp, 101
include/astNodeBinary.hpp, 102
include/astNodeBoolean.hpp, 103
include/astNodeCastBoolean.hpp, 104
include/astNodeCastFloat.hpp, 105
include/astNodeCastInteger.hpp, 106
include/astNodeFloat.hpp, 107
include/astNodeInteger.hpp, 108
include/astNodeNegative.hpp, 109
include/computedExpression.hpp, 110
include/computedExpressionBoolean.hpp, 111
include/computedExpressionError.hpp, 112
include/computedExpressionFloat.hpp, 113
include/computedExpressionInteger.hpp, 114
include/error.hpp, 115
include/garbageCollected.hpp, 116
include/macros.hpp, 116
include/opcode.hpp, 117
include/program.hpp, 118
include/singletonObjectPool.hpp, 120
include/tang.hpp, 121
include/tangBase.hpp, 122
include/tangScanner.hpp, 123
INTEGER
opcode.hpp, 118
is_equal
Tang::ComputedExpression, 40, 42
Tang::ComputedExpressionBoolean, 48, 49
Tang::ComputedExpressionError, 56, 57
Tang::ComputedExpressionFloat, 63, 64
Tang::ComputedExpressionInteger, 71, 72

location.hh
operator<<, 100, 101

macros.hpp
TANG_UNUSED, 117
make
Tang::GarbageCollected, 79
makeCopy
Tang::AstNode, 11
Tang::AstNodeBinary, 14
Tang::AstNodeBoolean, 17
Tang::AstNodeCastBoolean, 20
Tang::AstNodeCastFloat, 23
Tang::AstNodeCastInteger, 26
Tang::AstNodeFloat, 29
Tang::AstNodeInteger, 32
Tang::AstNodeNegative, 35
Tang::ComputedExpression, 42
Tang::ComputedExpressionBoolean, 49
Tang::ComputedExpressionError, 57
Tang::ComputedExpressionFloat, 65
Tang::ComputedExpressionInteger, 72
MODULO
opcode.hpp, 118
MULTIPLY
opcode.hpp, 118

NEGATIVE
opcode.hpp, 118

Opcode
opcode.hpp, 118
opcode.hpp
ADD, 118
BOOLEAN, 118
CASTBOOLEAN, 118
CASTFLOAT, 118
CASTINTEGER, 118
DIVIDE, 118
FLOAT, 118
INTEGER, 118
MODULO, 118

MULTIPLY, 118
NEGATIVE, 118
Opcode, 118
SUBTRACT, 118
operator<<
error.cpp, 136
location.hh, 100, 101
Tang::Error, 75
Tang::GarbageCollected, 86
operator*
Tang::GarbageCollected, 80
operator+
Tang::GarbageCollected, 81
operator-
Tang::GarbageCollected, 82
operator->
Tang::GarbageCollected, 83
operator/
Tang::GarbageCollected, 83
operator=
Tang::GarbageCollected, 84
operator==
Tang::GarbageCollected, 85, 86
operator%
Tang::GarbageCollected, 80

Program
Tang::Program, 92
program-dumpBytecode.cpp
DUMPPROGRAMCHECK, 138
program-execute.cpp
EXECUTEPROGRAMCHECK, 139
STACKCHECK, 139

recycle
Tang::SingletonObjectPool< T >, 95

Script
Tang::Program, 92
src/astNode.cpp, 124
src/astNodeBinary.cpp, 125
src/astNodeBoolean.cpp, 125
src/astNodeCastBoolean.cpp, 126
src/astNodeCastFloat.cpp, 127
src/astNodeCastInteger.cpp, 128
src/astNodeFloat.cpp, 129
src/astNodeInteger.cpp, 130
src/astNodeNegative.cpp, 131
src/computedExpression.cpp, 132
src/computedExpressionBoolean.cpp, 133
src/computedExpressionError.cpp, 134
src/computedExpressionFloat.cpp, 134
src/computedExpressionInteger.cpp, 135
src/error.cpp, 136
src/program-dumpBytecode.cpp, 137
src/program-execute.cpp, 138
src/program.cpp, 140
src/tangBase.cpp, 140
STACKCHECK
program-execute.cpp, 139
SUBTRACT
opcode.hpp, 118

Tang::AstNode, 9
AstNode, 11
makeCopy, 11
Tang::AstNodeBinary, 12
AstNodeBinary, 14
makeCopy, 14
Tang::AstNodeBoolean, 15
AstNodeBoolean, 17
makeCopy, 17
Tang::AstNodeCastBoolean, 18
AstNodeCastBoolean, 20
makeCopy, 20
Tang::AstNodeCastFloat, 21
AstNodeCastFloat, 23
makeCopy, 23
Tang::AstNodeCastInteger, 24
AstNodeCastInteger, 26
makeCopy, 26
Tang::AstNodeFloat, 27
AstNodeFloat, 29
makeCopy, 29
Tang::AstNodeInteger, 30
AstNodeInteger, 32
makeCopy, 32
Tang::AstNodeNegative, 33
AstNodeNegative, 35
makeCopy, 35
Tang::ComputedExpression, 36
__add, 37
__boolean, 37
__divide, 37
__float, 38
__integer, 38
__modulo, 38
__multiply, 39
__negative, 39
__subtract, 39
dump, 40
is_equal, 40, 42
makeCopy, 42
Tang::ComputedExpressionBoolean, 43
__add, 45
__boolean, 45
__divide, 45
__float, 46
__integer, 46
__modulo, 46
__multiply, 47
__negative, 47
__subtract, 47
ComputedExpressionBoolean, 44
dump, 48
is_equal, 48, 49
makeCopy, 49
Tang::ComputedExpressionError, 50

__add, 52
__boolean, 52
__divide, 53
__float, 53
__integer, 53
__modulo, 53
__multiply, 55
__negative, 55
__subtract, 55
ComputedExpressionError, 52
dump, 56
is_equal, 56, 57
makeCopy, 57
Tang::ComputedExpressionFloat, 58
__add, 60
__boolean, 60
__divide, 61
__float, 61
__integer, 61
__modulo, 62
__multiply, 62
__negative, 62
__subtract, 63
ComputedExpressionFloat, 60
dump, 63
is_equal, 63, 64
makeCopy, 65
Tang::ComputedExpressionInteger, 66
__add, 68
__boolean, 68
__divide, 68
__float, 69
__integer, 69
__modulo, 69
__multiply, 70
__negative, 70
__subtract, 70
ComputedExpressionInteger, 67
dump, 71
is_equal, 71, 72
makeCopy, 72
Tang::Error, 73
Error, 75
operator<<, 75
Tang::GarbageCollected, 76
~GarbageCollected, 79
GarbageCollected, 78, 79
make, 79
operator<<, 86
operator*, 80
operator+, 81
operator-, 82
operator->, 83
operator/, 83
operator=, 84
operator==, 85, 86
operator%, 80
Tang::location, 87
Tang::position, 88
Tang::Program, 90
addBytecode, 92
CodeType, 91
dumpBytecode, 92
execute, 93
getAst, 93
getCode, 93
getResult, 93
Program, 92
Script, 92
Template, 92
Tang::SingletonObjectPool< T >, 94
get, 94
getInstance, 95
recycle, 95
Tang::TangBase, 95
compileScript, 96
TangBase, 96
Tang::TangScanner, 97
get_next_token, 98
TangScanner, 98
TANG_UNUSED
macros.hpp, 117
TangBase
Tang::TangBase, 96
TangScanner
Tang::TangScanner, 98
Template
Tang::Program, 92
test/test.cpp, 141
test/testGarbageCollected.cpp, 142
test/testSingletonObjectPool.cpp, 143