

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	12
5.1.2 Constructor & Destructor Documentation	12
5.1.2.1 AstNode()	12
5.1.3 Member Function Documentation	12
5.1.3.1 makeCopy()	12
5.2 Tang::AstNodeAdd Class Reference	13
5.2.1 Detailed Description	15
5.2.2 Constructor & Destructor Documentation	15
5.2.2.1 AstNodeAdd()	15
5.2.3 Member Function Documentation	15
5.2.3.1 makeCopy()	15
5.3 Tang::AstNodeCastFloat Class Reference	16
5.3.1 Detailed Description	18
5.3.2 Constructor & Destructor Documentation	18
5.3.2.1 AstNodeCastFloat()	18
5.3.3 Member Function Documentation	18
5.3.3.1 makeCopy()	18
5.4 Tang::AstNodeCastInteger Class Reference	19
5.4.1 Detailed Description	21
5.4.2 Constructor & Destructor Documentation	21
5.4.2.1 AstNodeCastInteger()	21
5.4.3 Member Function Documentation	21
5.4.3.1 makeCopy()	21
5.5 Tang::AstNodeDivide Class Reference	22
5.5.1 Detailed Description	24
5.5.2 Constructor & Destructor Documentation	24
5.5.2.1 AstNodeDivide()	24
5.5.3 Member Function Documentation	24

5.5.3.1 makeCopy()	24
5.6 Tang::AstNodeFloat Class Reference	25
5.6.1 Detailed Description	27
5.6.2 Constructor & Destructor Documentation	27
5.6.2.1 AstNodeFloat()	27
5.6.3 Member Function Documentation	27
5.6.3.1 makeCopy()	27
5.7 Tang::AstNodeInteger Class Reference	28
5.7.1 Detailed Description	30
5.7.2 Constructor & Destructor Documentation	30
5.7.2.1 AstNodeInteger()	30
5.7.3 Member Function Documentation	30
5.7.3.1 makeCopy()	30
5.8 Tang::AstNodeModulo Class Reference	31
5.8.1 Detailed Description	33
5.8.2 Constructor & Destructor Documentation	33
5.8.2.1 AstNodeModulo()	33
5.8.3 Member Function Documentation	33
5.8.3.1 makeCopy()	33
5.9 Tang::AstNodeMultiply Class Reference	34
5.9.1 Detailed Description	36
5.9.2 Constructor & Destructor Documentation	36
5.9.2.1 AstNodeMultiply()	36
5.9.3 Member Function Documentation	36
5.9.3.1 makeCopy()	36
5.10 Tang::AstNodeNegative Class Reference	37
5.10.1 Detailed Description	39
5.10.2 Constructor & Destructor Documentation	39
5.10.2.1 AstNodeNegative()	39
5.10.3 Member Function Documentation	39
5.10.3.1 makeCopy()	39
5.11 Tang::AstNodeSubtract Class Reference	40
5.11.1 Detailed Description	42
5.11.2 Constructor & Destructor Documentation	42
5.11.2.1 AstNodeSubtract()	42
5.11.3 Member Function Documentation	42
5.11.3.1 makeCopy()	42
5.12 Tang::ComputedExpression Class Reference	43
5.12.1 Detailed Description	44
5.12.2 Member Function Documentation	44
5.12.2.1 __add()	44
5.12.2.2 __divide()	44

5.12.2.3 <code>__float()</code>	45
5.12.2.4 <code>__integer()</code>	45
5.12.2.5 <code>__modulo()</code>	45
5.12.2.6 <code>__multiply()</code>	46
5.12.2.7 <code>__negative()</code>	46
5.12.2.8 <code>__subtract()</code>	46
5.12.2.9 <code>dump()</code>	47
5.12.2.10 <code>is_equal()</code> [1/3]	47
5.12.2.11 <code>is_equal()</code> [2/3]	47
5.12.2.12 <code>is_equal()</code> [3/3]	48
5.12.2.13 <code>makeCopy()</code>	48
5.13 Tang::ComputedExpressionError Class Reference	48
5.13.1 Detailed Description	50
5.13.2 Constructor & Destructor Documentation	50
5.13.2.1 <code>ComputedExpressionError()</code>	50
5.13.3 Member Function Documentation	50
5.13.3.1 <code>__add()</code>	50
5.13.3.2 <code>__divide()</code>	51
5.13.3.3 <code>__float()</code>	51
5.13.3.4 <code>__integer()</code>	52
5.13.3.5 <code>__modulo()</code>	52
5.13.3.6 <code>__multiply()</code>	52
5.13.3.7 <code>__negative()</code>	53
5.13.3.8 <code>__subtract()</code>	53
5.13.3.9 <code>dump()</code>	53
5.13.3.10 <code>is_equal()</code> [1/3]	53
5.13.3.11 <code>is_equal()</code> [2/3]	54
5.13.3.12 <code>is_equal()</code> [3/3]	54
5.13.3.13 <code>makeCopy()</code>	55
5.14 Tang::ComputedExpressionFloat Class Reference	55
5.14.1 Detailed Description	57
5.14.2 Constructor & Destructor Documentation	57
5.14.2.1 <code>ComputedExpressionFloat()</code>	57
5.14.3 Member Function Documentation	57
5.14.3.1 <code>__add()</code>	57
5.14.3.2 <code>__divide()</code>	57
5.14.3.3 <code>__float()</code>	58
5.14.3.4 <code>__integer()</code>	58
5.14.3.5 <code>__modulo()</code>	58
5.14.3.6 <code>__multiply()</code>	59
5.14.3.7 <code>__negative()</code>	59
5.14.3.8 <code>__subtract()</code>	59

5.14.3.9 dump()	60
5.14.3.10 is_equal() [1/3]	60
5.14.3.11 is_equal() [2/3]	60
5.14.3.12 is_equal() [3/3]	61
5.14.3.13 makeCopy()	61
5.15 Tang::ComputedExpressionInteger Class Reference	62
5.15.1 Detailed Description	63
5.15.2 Constructor & Destructor Documentation	63
5.15.2.1 ComputedExpressionInteger()	63
5.15.3 Member Function Documentation	64
5.15.3.1 __add()	64
5.15.3.2 __divide()	64
5.15.3.3 __float()	64
5.15.3.4 __integer()	65
5.15.3.5 __modulo()	65
5.15.3.6 __multiply()	65
5.15.3.7 __negative()	66
5.15.3.8 __subtract()	66
5.15.3.9 dump()	66
5.15.3.10 is_equal() [1/3]	67
5.15.3.11 is_equal() [2/3]	67
5.15.3.12 is_equal() [3/3]	67
5.15.3.13 makeCopy()	68
5.16 Tang::Error Class Reference	69
5.16.1 Detailed Description	70
5.16.2 Constructor & Destructor Documentation	70
5.16.2.1 Error() [1/2]	70
5.16.2.2 Error() [2/2]	70
5.16.3 Friends And Related Function Documentation	70
5.16.3.1 operator<<	71
5.17 Tang::GarbageCollected Class Reference	71
5.17.1 Detailed Description	73
5.17.2 Constructor & Destructor Documentation	73
5.17.2.1 GarbageCollected() [1/3]	73
5.17.2.2 GarbageCollected() [2/3]	73
5.17.2.3 ~GarbageCollected()	74
5.17.2.4 GarbageCollected() [3/3]	74
5.17.3 Member Function Documentation	74
5.17.3.1 make()	74
5.17.3.2 operator%()	75
5.17.3.3 operator*() [1/2]	75
5.17.3.4 operator*() [2/2]	76

5.17.3.5 operator+()	76
5.17.3.6 operator-() [1/2]	77
5.17.3.7 operator-() [2/2]	77
5.17.3.8 operator->()	78
5.17.3.9 operator/()	78
5.17.3.10 operator=() [1/2]	79
5.17.3.11 operator=() [2/2]	79
5.17.3.12 operator==() [1/3]	80
5.17.3.13 operator==() [2/3]	80
5.17.3.14 operator==() [3/3]	81
5.17.4 Friends And Related Function Documentation	81
5.17.4.1 operator<<	81
5.18 Tang::location Class Reference	81
5.18.1 Detailed Description	83
5.19 Tang::position Class Reference	83
5.19.1 Detailed Description	84
5.20 Tang::Program Class Reference	84
5.20.1 Detailed Description	86
5.20.2 Member Enumeration Documentation	86
5.20.2.1 CodeType	86
5.20.3 Constructor & Destructor Documentation	86
5.20.3.1 Program()	86
5.20.4 Member Function Documentation	86
5.20.4.1 addBytecode()	87
5.20.4.2 dumpBytecode()	87
5.20.4.3 execute()	87
5.20.4.4 getAst()	87
5.20.4.5 getCode()	88
5.20.4.6 getResult()	88
5.21 Tang::SingletonObjectPool< T > Class Template Reference	88
5.21.1 Member Function Documentation	88
5.21.1.1 get()	89
5.21.1.2 getInstance()	89
5.21.1.3 recycle()	89
5.22 Tang::TangBase Class Reference	89
5.22.1 Detailed Description	90
5.22.2 Constructor & Destructor Documentation	90
5.22.2.1 TangBase()	90
5.22.3 Member Function Documentation	90
5.22.3.1 compileScript()	90
5.23 Tang::TangScanner Class Reference	91
5.23.1 Detailed Description	92

5.23.2 Constructor & Destructor Documentation	92
5.23.2.1 TangScanner()	92
5.23.3 Member Function Documentation	92
5.23.3.1 get_next_token()	92
6 File Documentation	93
6.1 build/generated/location.hh File Reference	93
6.1.1 Detailed Description	94
6.1.2 Function Documentation	94
6.1.2.1 operator<<() [1/2]	94
6.1.2.2 operator<<() [2/2]	95
6.2 include/astNode.hpp File Reference	95
6.2.1 Detailed Description	96
6.3 include/astNodeAdd.hpp File Reference	96
6.4 include/astNodeCastFloat.hpp File Reference	97
6.5 include/astNodeCastInteger.hpp File Reference	98
6.6 include/astNodeDivide.hpp File Reference	99
6.7 include/astNodeFloat.hpp File Reference	100
6.8 include/astNodeInteger.hpp File Reference	101
6.9 include/astNodeModulo.hpp File Reference	102
6.10 include/astNodeMultiply.hpp File Reference	103
6.11 include/astNodeNegative.hpp File Reference	104
6.12 include/astNodeSubtract.hpp File Reference	105
6.13 include/computedExpression.hpp File Reference	106
6.14 include/computedExpressionError.hpp File Reference	107
6.15 include/computedExpressionFloat.hpp File Reference	108
6.16 include/computedExpressionInteger.hpp File Reference	109
6.17 include/error.hpp File Reference	109
6.17.1 Detailed Description	110
6.18 include/garbageCollected.hpp File Reference	110
6.19 include/macros.hpp File Reference	111
6.19.1 Detailed Description	111
6.19.2 Macro Definition Documentation	112
6.19.2.1 TANG_UNUSED	112
6.20 include/opcode.hpp File Reference	112
6.20.1 Detailed Description	112
6.20.2 Enumeration Type Documentation	112
6.20.2.1 Opcode	112
6.21 include/program.hpp File Reference	113
6.21.1 Detailed Description	114
6.22 include/singletonObjectPool.hpp File Reference	114
6.23 include/tang.hpp File Reference	115

6.23.1 Detailed Description	116
6.24 include/tangBase.hpp File Reference	116
6.24.1 Detailed Description	117
6.25 include/tangScanner.hpp File Reference	117
6.25.1 Detailed Description	118
6.26 src/astNode.cpp File Reference	118
6.27 src/astNodeAdd.cpp File Reference	119
6.28 src/astNodeCastFloat.cpp File Reference	119
6.29 src/astNodeCastInteger.cpp File Reference	120
6.30 src/astNodeDivide.cpp File Reference	121
6.31 src/astNodeFloat.cpp File Reference	122
6.32 src/astNodeInteger.cpp File Reference	123
6.33 src/astNodeModulo.cpp File Reference	124
6.34 src/astNodeMultiply.cpp File Reference	125
6.35 src/astNodeNegative.cpp File Reference	126
6.36 src/astNodeSubtract.cpp File Reference	127
6.37 src/computedExpression.cpp File Reference	128
6.38 src/computedExpressionError.cpp File Reference	129
6.39 src/computedExpressionFloat.cpp File Reference	129
6.40 src/computedExpressionInteger.cpp File Reference	130
6.41 src/error.cpp File Reference	131
6.41.1 Function Documentation	131
6.41.1.1 operator<<()	131
6.42 src/program-dumpBytecode.cpp File Reference	132
6.42.1 Macro Definition Documentation	132
6.42.1.1 DUMPPROGRAMCHECK	132
6.43 src/program-execute.cpp File Reference	133
6.43.1 Macro Definition Documentation	133
6.43.1.1 EXECUTEPROGRAMCHECK	133
6.43.1.2 STACKCHECK	134
6.44 src/program.cpp File Reference	134
6.45 src/tangBase.cpp File Reference	135
6.46 test/test.cpp File Reference	135
6.46.1 Detailed Description	136
6.47 test/testSingletonObjectPool.cpp File Reference	136
Index	137

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::AstNodeAdd	13
Tang::AstNodeCastFloat	16
Tang::AstNodeCastInteger	19
Tang::AstNodeDivide	22
Tang::AstNodeFloat	25
Tang::AstNodeInteger	28
Tang::AstNodeModulo	31
Tang::AstNodeMultiply	34
Tang::AstNodeNegative	37
Tang::AstNodeSubtract	40
Tang::ComputedExpression	43
Tang::ComputedExpressionError	48
Tang::ComputedExpressionFloat	55
Tang::ComputedExpressionInteger	62
Tang::Error	69
Tang::GarbageCollected	71
Tang::location	81
Tang::position	83
Tang::Program	84
Tang::SingletonObjectPool< T >	88
Tang::TangBase	89
TangTangFlexLexer	
Tang::TangScanner	91

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::AstNodeAdd	An AstNode that represents a "+" expression	13
Tang::AstNodeCastFloat	An AstNode that represents a typecast to a float	16
Tang::AstNodeCastInteger	An AstNode that represents a typecast to an integer	19
Tang::AstNodeDivide	An AstNode that represents a "/" expression	22
Tang::AstNodeFloat	An AstNode that represents an float literal	25
Tang::AstNodeInteger	An AstNode that represents an integer literal	28
Tang::AstNodeModulo	An AstNode that represents a "%" expression	31
Tang::AstNodeMultiply	An AstNode that represents a "*" expression	34
Tang::AstNodeNegative	An AstNode that represents a unary negation	37
Tang::AstNodeSubtract	An AstNode that represents a "-" expression	40
Tang::ComputedExpression	Represents the result of a computation that has been executed	43
Tang::ComputedExpressionError	Represents a Runtime Error	48
Tang::ComputedExpressionFloat	Represents a Float that is the result of a computation	55
Tang::ComputedExpressionInteger	Represents an Integer that is the result of a computation	62
Tang::Error	Used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error	69
Tang::GarbageCollected	A container that acts as a resource-counting garbage collector for the specified type	71

Tang::location	
Two points in a source file	81
Tang::position	
A point in a source file	83
Tang::Program	
Represents a compiled script or template that may be executed	84
Tang::SingletonObjectPool< T >	88
Tang::TangBase	
The base class for the Tang programming language	89
Tang::TangScanner	
The Flex lexer class for the main Tang language	91

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	93
include/astNode.hpp	
Define the Tang::AstNode and its associated/derivative classes	95
include/astNodeAdd.hpp	96
include/astNodeCastFloat.hpp	97
include/astNodeCastInteger.hpp	98
include/astNodeDivide.hpp	99
include/astNodeFloat.hpp	100
include/astNodeInteger.hpp	101
include/astNodeModulo.hpp	102
include/astNodeMultiply.hpp	103
include/astNodeNegative.hpp	104
include/astNodeSubtract.hpp	105
include/computedExpression.hpp	106
include/computedExpressionError.hpp	107
include/computedExpressionFloat.hpp	108
include/computedExpressionInteger.hpp	109
include/error.hpp	
Define the Tang::Error class used to describe syntax and runtime errors	109
include/garbageCollected.hpp	110
include/macros.hpp	
Contains generic macros	111
include/opcode.hpp	
Declare the Opcodes used in the Bytecode representation of a program	112
include/program.hpp	
Define the Tang::Program class used to compile and execute source code	113
include/singletonObjectPool.hpp	114
include/tang.hpp	
Header file supplied for use by 3rd party code so that they can easily include all necessary headers	115
include/tangBase.hpp	
Defines the Tang::TangBase class used to interact with Tang	116
include/tangScanner.hpp	
Defines the Tang::TangScanner used to tokenize a Tang script	117

src/ astNode.cpp	118
src/ astNodeAdd.cpp	119
src/ astNodeCastFloat.cpp	119
src/ astNodeCastInteger.cpp	120
src/ astNodeDivide.cpp	121
src/ astNodeFloat.cpp	122
src/ astNodeInteger.cpp	123
src/ astNodeModulo.cpp	124
src/ astNodeMultiply.cpp	125
src/ astNodeNegative.cpp	126
src/ astNodeSubtract.cpp	127
src/ computedExpression.cpp	128
src/ computedExpressionError.cpp	129
src/ computedExpressionFloat.cpp	129
src/ computedExpressionInteger.cpp	130
src/ error.cpp	131
src/ program-dumpBytecode.cpp	132
src/ program-execute.cpp	133
src/ program.cpp	134
src/ tangBase.cpp	135
test/ test.cpp	
Test the general language behaviors	135
test/ testSingletonObjectPool.cpp	136

Chapter 5

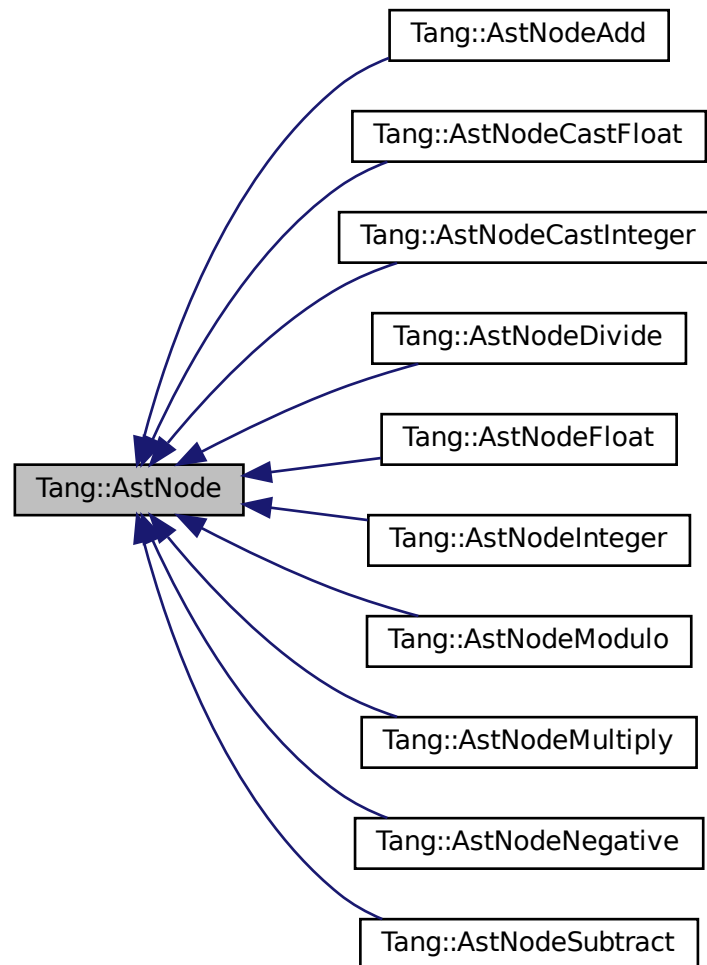
Class Documentation

5.1 Tang::AstNode Class Reference

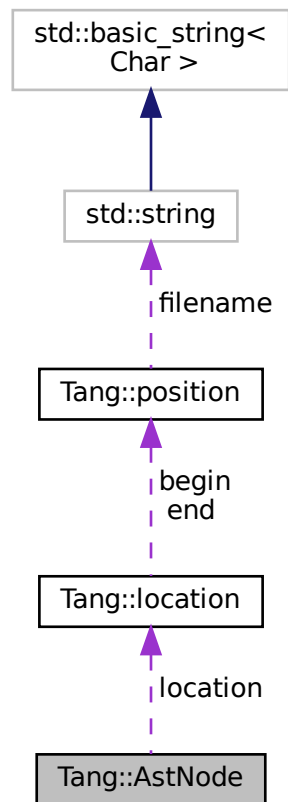
Base class for representing nodes of an Abstract Syntax Tree (AST).

```
#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 `AstNode * makeCopy` () const
Provide a copy of the `AstNode` (recursively, if appropriate).

Protected Member Functions

- `AstNode` (`Tang::location loc`)
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()

```
Tang::AstNode::AstNode (
    Tang::location loc ) [inline], [protected]
```

The generic constructor.

It should never be called on its own.

Parameters

<i>loc</i>	The location associated with this node.
------------	---

5.1.3 Member Function Documentation

5.1.3.1 makeCopy()

```
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::AstNodeSubtract](#), [Tang::AstNodeNegative](#), [Tang::AstNodeMultiply](#), [Tang::AstNodeModulo](#), [Tang::AstNodeInteger](#), [Tang::AstNodeFloat](#), [Tang::AstNodeDivide](#), [Tang::AstNodeCastInteger](#), [Tang::AstNodeCastFloat](#), and [Tang::AstNodeAdd](#).

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

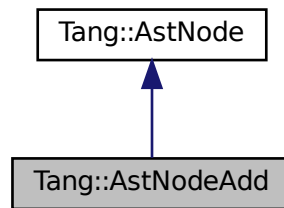
- include/[astNode.hpp](#)
- src/[astNode.cpp](#)

5.2 Tang::AstNodeAdd Class Reference

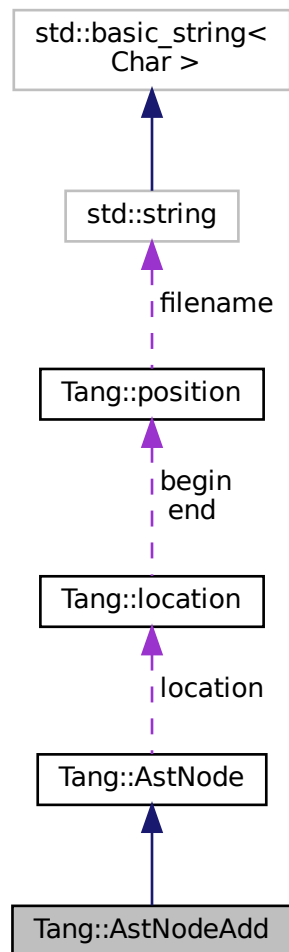
An [AstNode](#) that represents a "+" expression.

```
#include <astNodeAdd.hpp>
```

Inheritance diagram for Tang::AstNodeAdd:



Collaboration diagram for Tang::AstNodeAdd:



Public Member Functions

- `AstNodeAdd` (`AstNode` *lhs, `AstNode` *rhs, `Tang::location` loc)
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 `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 "+" expression.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AstNodeAdd()

```
Tang::AstNodeAdd::AstNodeAdd (
    AstNode * lhs,
    AstNode * rhs,
    Tang::location loc ) [inline]
```

The constructor.

Parameters

<i>lhs</i>	The left hand side expression.
<i>rhs</i>	The right hand side expression.
<i>loc</i>	The location associated with the expression. @location The location associated with this node.

5.2.3 Member Function Documentation

5.2.3.1 makeCopy()

```
AstNode * AstNodeAdd::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:

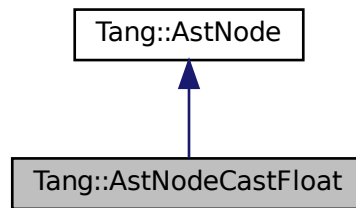
- include/[astNodeAdd.hpp](#)
- src/[astNodeAdd.cpp](#)

5.3 Tang::AstNodeCastFloat Class Reference

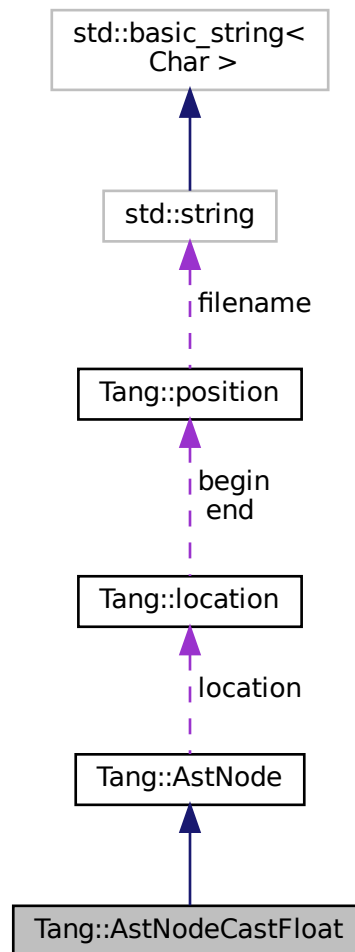
An [AstNode](#) that represents a typecast to a float.

```
#include <astNodeCastFloat.hpp>
```

Inheritance diagram for Tang::AstNodeCastFloat:



Collaboration diagram for Tang::AstNodeCastFloat:



Public Member Functions

- [AstNodeCastFloat](#) ([AstNode](#) *expression, [Tang::location](#) loc)
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 [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 typecast to a float.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 AstNodeCastFloat()

```
Tang::AstNodeCastFloat::AstNodeCastFloat (
    AstNode * expression,
    Tang::location loc ) [inline]
```

The constructor.

Parameters

<i>expression</i>	The expression to be typecast. @location The location associated with this node.
-------------------	--

5.3.3 Member Function Documentation

5.3.3.1 makeCopy()

```
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:

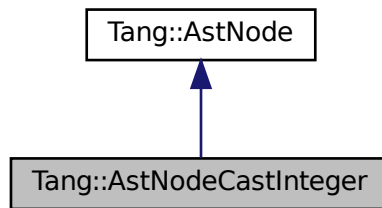
- [include/astNodeCastFloat.hpp](#)
- [src/astNodeCastFloat.cpp](#)

5.4 Tang::AstNodeCastInteger Class Reference

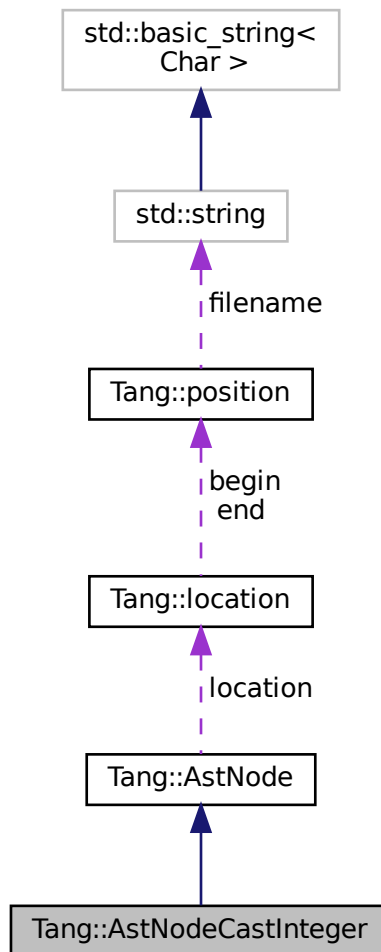
An [AstNode](#) that represents a typecast to an integer.

```
#include <astNodeCastInteger.hpp>
```

Inheritance diagram for Tang::AstNodeCastInteger:



Collaboration diagram for Tang::AstNodeCastInteger:



Public Member Functions

- `AstNodeCastInteger` (`AstNode` *expression, `Tang::location` loc)
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 `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 an integer.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 AstNodeCastInteger()

```
Tang::AstNodeCastInteger::AstNodeCastInteger (
    AstNode * expression,
    Tang::location loc ) [inline]
```

The constructor.

Parameters

<i>expression</i>	The expression to be typecast. @location The location associated with this node.
-------------------	--

5.4.3 Member Function Documentation

5.4.3.1 makeCopy()

```
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:

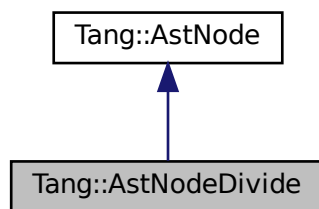
- [include/astNodeCastInteger.hpp](#)
- [src/astNodeCastInteger.cpp](#)

5.5 Tang::AstNodeDivide Class Reference

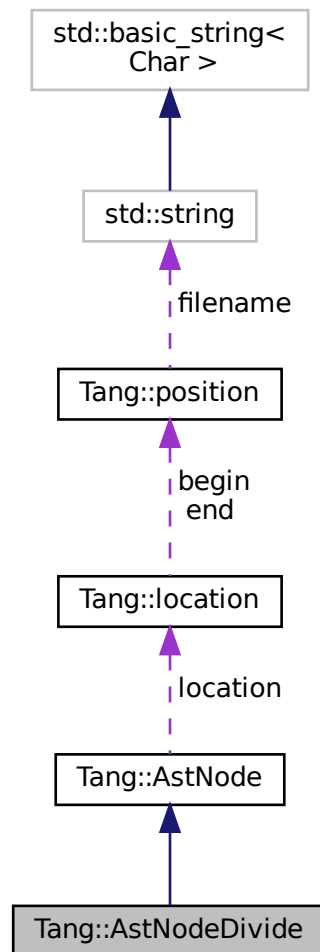
An [AstNode](#) that represents a "/" expression.

```
#include <astNodeDivide.hpp>
```

Inheritance diagram for Tang::AstNodeDivide:



Collaboration diagram for Tang::AstNodeDivide:



Public Member Functions

- [AstNodeDivide](#) ([AstNode](#) *lhs, [AstNode](#) *rhs, [Tang::location](#) loc)
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 [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 "/" expression.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 AstNodeDivide()

```
Tang::AstNodeDivide::AstNodeDivide (
    AstNode * lhs,
    AstNode * rhs,
    Tang::location loc ) [inline]
```

The constructor.

Parameters

<i>lhs</i>	The left hand side expression.
<i>rhs</i>	The right hand side expression.
<i>loc</i>	The location associated with the expression. @location The location associated with this node.

5.5.3 Member Function Documentation

5.5.3.1 makeCopy()

```
AstNode * AstNodeDivide::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:

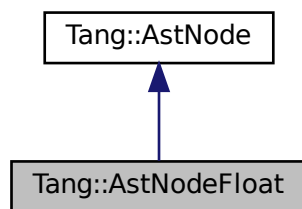
- include/[astNodeDivide.hpp](#)
- src/[astNodeDivide.cpp](#)

5.6 Tang::AstNodeFloat Class Reference

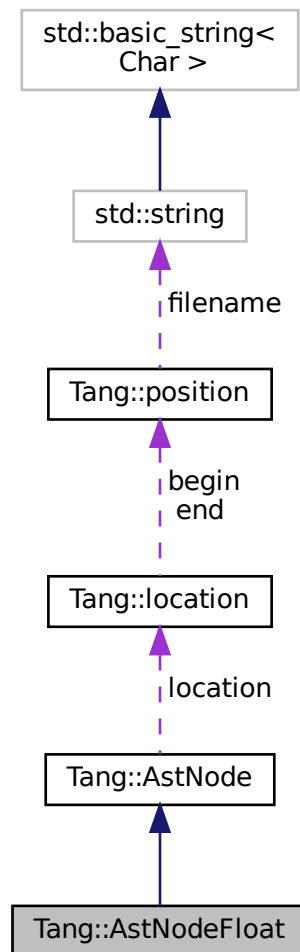
An [AstNode](#) that represents an float literal.

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

Inheritance diagram for Tang::AstNodeFloat:



Collaboration diagram for Tang::AstNodeFloat:



Public Member Functions

- [AstNodeFloat](#) (double number, [Tang::location](#) loc)
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 [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 an float literal.

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

5.6.2 Constructor & Destructor Documentation

5.6.2.1 AstNodeFloat()

```
Tang::AstNodeFloat::AstNodeFloat (
    double number,
    Tang::location loc ) [inline]
```

The constructor.

Parameters

<i>number</i>	The number to represent.
<i>loc</i>	The location associated with the expression. @location The location associated with this node.

5.6.3 Member Function Documentation

5.6.3.1 makeCopy()

```
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:

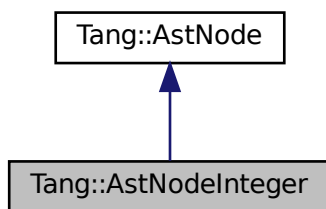
- include/[astNodeFloat.hpp](#)
- src/[astNodeFloat.cpp](#)

5.7 Tang::AstNodeInteger Class Reference

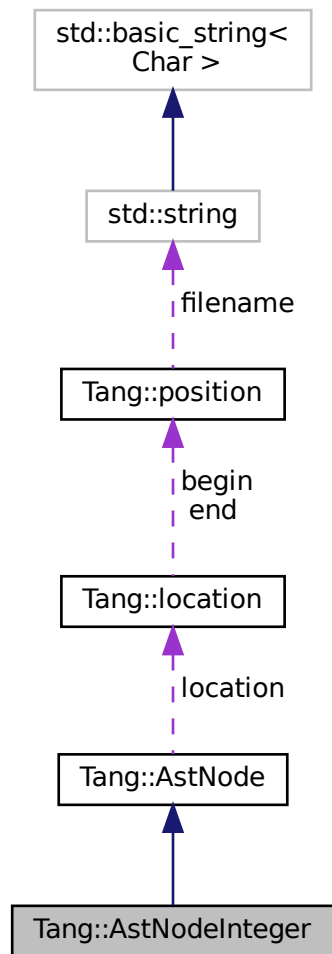
An [AstNode](#) that represents an integer literal.

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

Inheritance diagram for Tang::AstNodeInteger:



Collaboration diagram for Tang::AstNodeInteger:



Public Member Functions

- [AstNodeInteger](#) (int64_t number, [Tang::location](#) loc)
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 [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 integer literal.

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

5.7.2 Constructor & Destructor Documentation

5.7.2.1 AstNodeInteger()

```
Tang::AstNodeInteger::AstNodeInteger (
    int64_t number,
    Tang::location loc ) [inline]
```

The constructor.

Parameters

<i>number</i>	The number to represent.
<i>loc</i>	The location associated with the expression. @location The location associated with this node.

5.7.3 Member Function Documentation

5.7.3.1 makeCopy()

```
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:

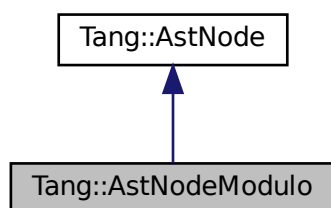
- include/[astNodeInteger.hpp](#)
- src/[astNodeInteger.cpp](#)

5.8 Tang::AstNodeModulo Class Reference

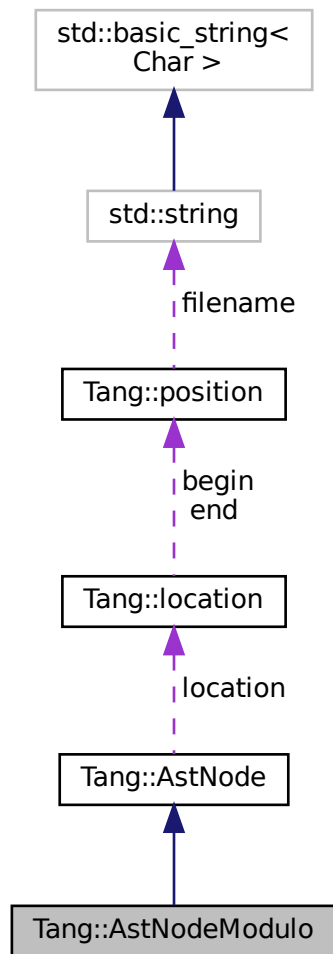
An [AstNode](#) that represents a "%" expression.

```
#include <astNodeModulo.hpp>
```

Inheritance diagram for Tang::AstNodeModulo:



Collaboration diagram for Tang::AstNodeModulo:



Public Member Functions

- `AstNodeModulo` (`AstNode` *lhs, `AstNode` *rhs, `Tang::location` loc)
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 `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 a "%" expression.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 AstNodeModulo()

```
Tang::AstNodeModulo::AstNodeModulo (
    AstNode * lhs,
    AstNode * rhs,
    Tang::location loc ) [inline]
```

The constructor.

Parameters

<i>lhs</i>	The left hand side expression.
<i>rhs</i>	The right hand side expression.
<i>loc</i>	The location associated with the expression. @location The location associated with this node.

5.8.3 Member Function Documentation

5.8.3.1 makeCopy()

```
AstNode * AstNodeModulo::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:

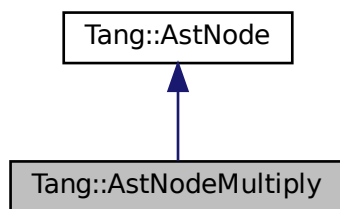
- include/[astNodeModulo.hpp](#)
- src/[astNodeModulo.cpp](#)

5.9 Tang::AstNodeMultiply Class Reference

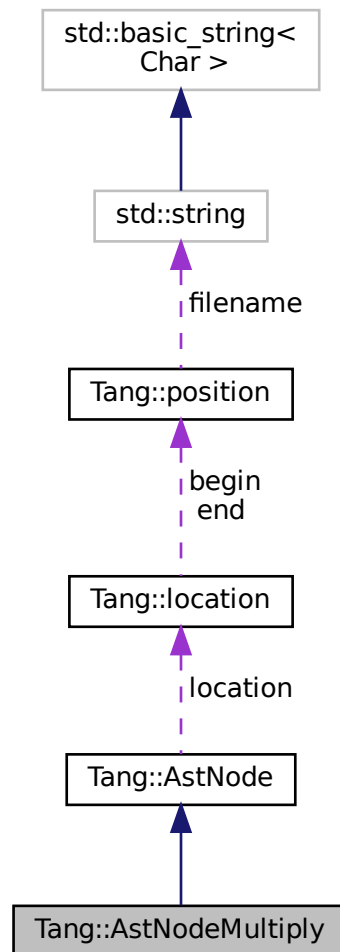
An [AstNode](#) that represents a "*" expression.

```
#include <astNodeMultiply.hpp>
```

Inheritance diagram for Tang::AstNodeMultiply:



Collaboration diagram for Tang::AstNodeMultiply:



Public Member Functions

- [AstNodeMultiply](#) ([AstNode](#) *lhs, [AstNode](#) *rhs, [Tang::location](#) loc)
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 [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 "*" expression.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 AstNodeMultiply()

```
Tang::AstNodeMultiply::AstNodeMultiply (
    AstNode * lhs,
    AstNode * rhs,
    Tang::location loc ) [inline]
```

The constructor.

Parameters

<i>lhs</i>	The left hand side expression.
<i>rhs</i>	The right hand side expression.
<i>loc</i>	The location associated with the expression. @location The location associated with this node.

5.9.3 Member Function Documentation

5.9.3.1 makeCopy()

```
AstNode * AstNodeMultiply::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:

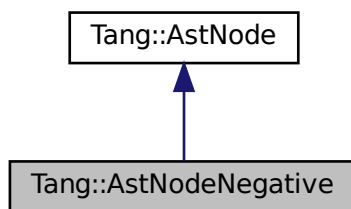
- include/[astNodeMultiply.hpp](#)
- src/[astNodeMultiply.cpp](#)

5.10 Tang::AstNodeNegative Class Reference

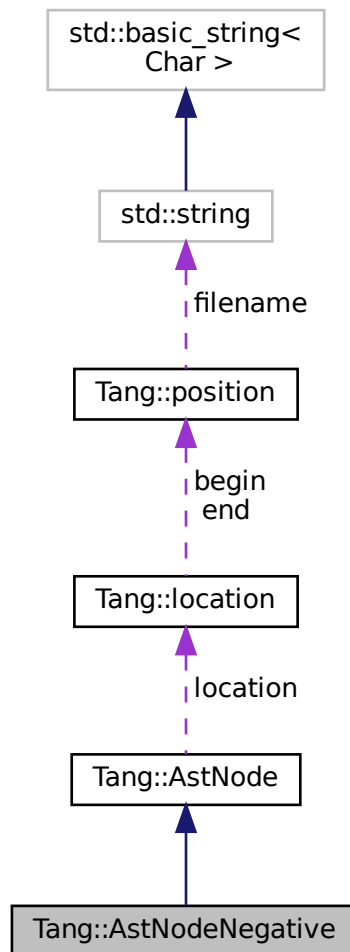
An [AstNode](#) that represents a unary negation.

```
#include <astNodeNegative.hpp>
```

Inheritance diagram for Tang::AstNodeNegative:



Collaboration diagram for Tang::AstNodeNegative:



Public Member Functions

- `AstNodeNegative` (`AstNode` *operand, `Tang::location` loc)
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 `AstNode` * `makeCopy` () const override
Provide a copy of the `AstNode` (recursively, if appropriate).

Protected Attributes

- `Tang::location` location
The location associated with this node.

5.10.1 Detailed Description

An [AstNode](#) that represents a unary negation.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 AstNodeNegative()

```
Tang::AstNodeNegative::AstNodeNegative (
    AstNode * operand,
    Tang::location loc ) [inline]
```

The constructor.

Parameters

<i>operand</i>	The expression to negate.
<i>loc</i>	The location associated with the expression.

5.10.3 Member Function Documentation

5.10.3.1 makeCopy()

```
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:

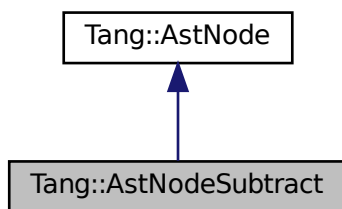
- include/[astNodeNegative.hpp](#)
- src/[astNodeNegative.cpp](#)

5.11 Tang::AstNodeSubtract Class Reference

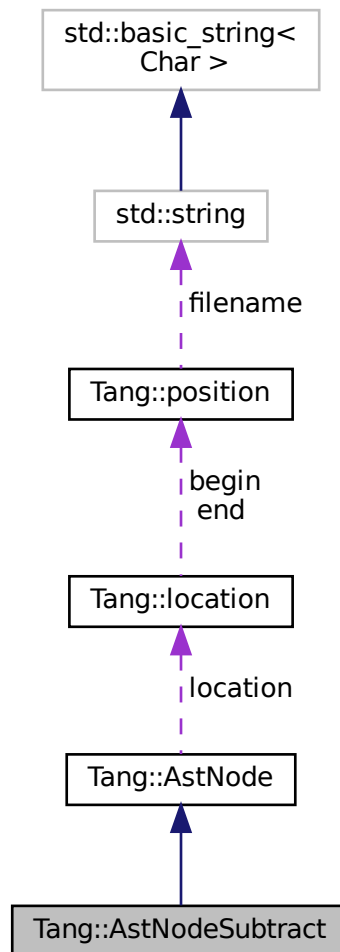
An [AstNode](#) that represents a "-" expression.

```
#include <astNodeSubtract.hpp>
```

Inheritance diagram for Tang::AstNodeSubtract:



Collaboration diagram for Tang::AstNodeSubtract:



Public Member Functions

- `AstNodeSubtract (AstNode *lhs, AstNode *rhs, Tang::location loc)`
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 `AstNode * makeCopy ()` const override
Provide a copy of the AstNode (recursively, if appropriate).

Protected Attributes

- `Tang::location location`
The location associated with this node.

5.11.1 Detailed Description

An [AstNode](#) that represents a "-" expression.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 AstNodeSubtract()

```
Tang::AstNodeSubtract::AstNodeSubtract (
    AstNode * lhs,
    AstNode * rhs,
    Tang::location loc ) [inline]
```

The constructor.

Parameters

<i>lhs</i>	The left hand side expression.
<i>rhs</i>	The right hand side expression.
<i>loc</i>	The location associated with the expression. @location The location associated with this node.

5.11.3 Member Function Documentation

5.11.3.1 makeCopy()

```
AstNode * AstNodeSubtract::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:

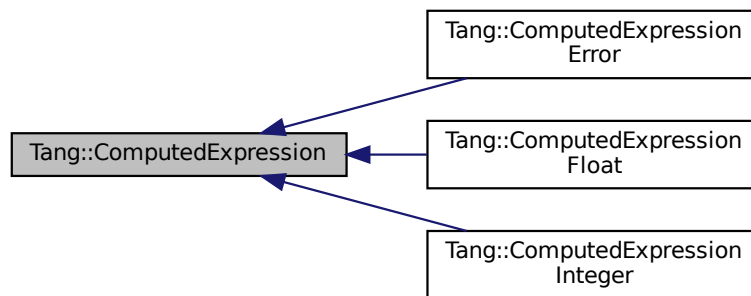
- include/[astNodeSubtract.hpp](#)
- src/[astNodeSubtract.cpp](#)

5.12 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 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.

5.12.1 Detailed Description

Represents the result of a computation that has been executed.

5.12.2 Member Function Documentation

5.12.2.1 `__add()`

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to add to this.
------------	--

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.12.2.2 `__divide()`

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to divide this by.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.12.2.3 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.12.2.4 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.12.2.5 __modulo()

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to modulo this by.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#).

5.12.2.6 `__multiply()`

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to multiply to this.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.12.2.7 `__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](#).

5.12.2.8 `__subtract()`

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to subtract from this.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.12.2.9 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](#), and [Tang::ComputedExpressionError](#).

5.12.2.10 is_equal() [1/3]

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

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

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.12.2.11 is_equal() [2/3]

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

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

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionError](#).

5.12.2.12 `is_equal()` [3/3]

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

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

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.12.2.13 `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](#), and [Tang::ComputedExpressionError](#).

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

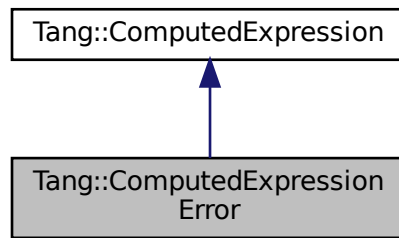
- [include/computedExpression.hpp](#)
- [src/computedExpression.cpp](#)

5.13 [Tang::ComputedExpressionError](#) Class Reference

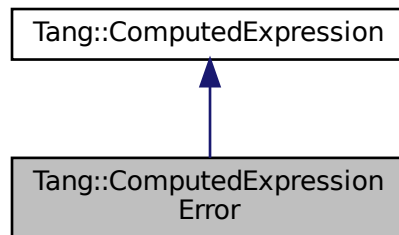
Represents a Runtime [Error](#).

```
#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 [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.

5.13.1 Detailed Description

Represents a Runtime `Error`.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 ComputedExpressionError()

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

Construct a Runtime `Error`.

Parameters

<code>error</code>	The <code>Tang::Error</code> object.
--------------------	--------------------------------------

5.13.3 Member Function Documentation

5.13.3.1 __add()

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to add to this.
------------	--

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.13.3.2 __divide()

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to divide this by.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.13.3.3 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.13.3.4 `__integer()`

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.13.3.5 `__modulo()`

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to modulo this by.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#).

5.13.3.6 `__multiply()`

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to multiply to this.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.13.3.7 __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](#).

5.13.3.8 __subtract()

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to subtract from this.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.13.3.9 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](#).

5.13.3.10 is_equal() [1/3]

```
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

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.13.3.11 is_equal() [2/3]

```
bool ComputedExpressionError::is_equal (
    const Error & val ) const  [override], [virtual]
```

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

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented from [Tang::ComputedExpression](#).

5.13.3.12 is_equal() [3/3]

```
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

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.13.3.13 makeCopy()

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

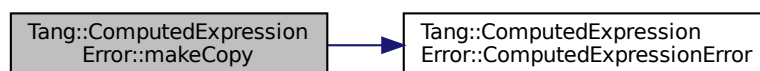
Make a copy of the [ComputedExpression](#) (recursively, if appropriate).

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:

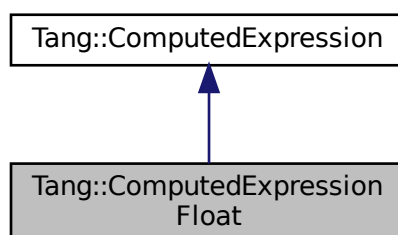
- [include/computedExpressionError.hpp](#)
- [src/computedExpressionError.cpp](#)

5.14 Tang::ComputedExpressionFloat Class Reference

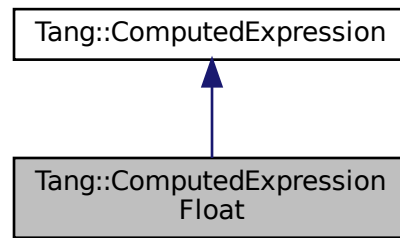
Represents a Float that is the result of a computation.

```
#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 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](#)

5.14.1 Detailed Description

Represents a Float that is the result of a computation.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 ComputedExpressionFloat()

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

Construct a Float result.

Parameters

<i>val</i>	The float value.
------------	------------------

5.14.3 Member Function Documentation

5.14.3.1 __add()

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to add to this.
------------	--

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.2 __divide()

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to divide this by.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.3 `__float()`

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.4 `__integer()`

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.5 `__modulo()`

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to modulo this by.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#).

5.14.3.6 __multiply()

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to multiply to this.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.7 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.8 __subtract()

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to subtract from this.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.9 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](#).

5.14.3.10 is_equal() [1/3]

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

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

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.11 is_equal() [2/3]

```
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

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionError](#).

5.14.3.12 is_equal() [3/3]

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

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

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.13 makeCopy()

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

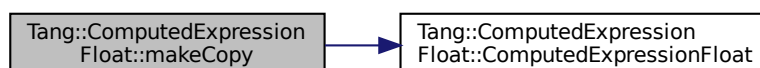
Make a copy of the [ComputedExpression](#) (recursively, if appropriate).

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:

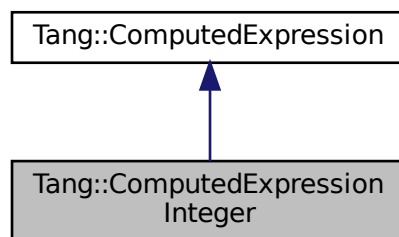
- [include/computedExpressionFloat.hpp](#)
- [src/computedExpressionFloat.cpp](#)

5.15 Tang::ComputedExpressionInteger Class Reference

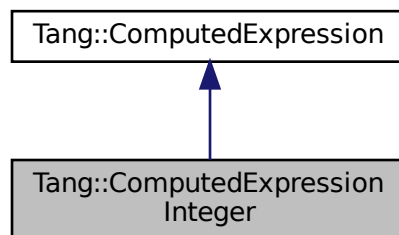
Represents an Integer that is the result of a computation.

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

Friends

- class `ComputedExpressionFloat`

5.15.1 Detailed Description

Represents an Integer that is the result of a computation.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 ComputedExpressionInteger()

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

Construct an Integer result.

Parameters

<i>val</i>	The integer value.
------------	--------------------

5.15.3 Member Function Documentation

5.15.3.1 `__add()`

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to add to this.
------------	--

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.2 `__divide()`

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to divide this by.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.3 `__float()`

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.4 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.5 __modulo()

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to modulo this by.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.6 __multiply()

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to multiply to this.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.7 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.8 __subtract()

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

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

Parameters

<i>rhs</i>	The GarbageCollected value to subtract from this.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.9 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](#).

5.15.3.10 is_equal() [1/3]

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

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

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.11 is_equal() [2/3]

```
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

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionError](#).

5.15.3.12 is_equal() [3/3]

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

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

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.13 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:

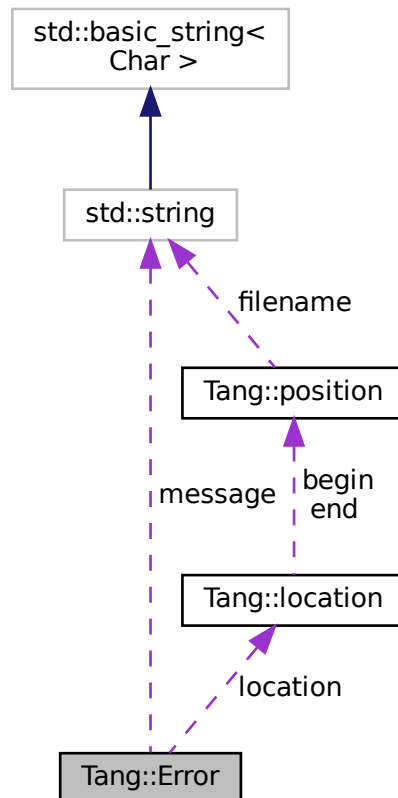
- [include/computedExpressionInteger.hpp](#)
- [src/computedExpressionInteger.cpp](#)

5.16 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.

5.16.1 Detailed Description

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

5.16.2 Constructor & Destructor Documentation

5.16.2.1 [Error\(\)](#) [1/2]

```
Tang::Error::Error (  
    std::string message ) [inline]
```

Creates an error message using the supplied error string and location.

Parameters

<i>message</i>	The error message as a string.
----------------	--------------------------------

5.16.2.2 [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

<i>message</i>	The error message as a string.
<i>location</i>	The location of the error.

5.16.3 Friends And Related Function Documentation

5.16.3.1 operator<<

```
std::ostream& operator<< (
    std::ostream & out,
    const Error & error ) [friend]
```

Add friendly output.

Parameters

<i>out</i>	The output stream.
<i>error</i>	The Error object.

Returns

The output stream.

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

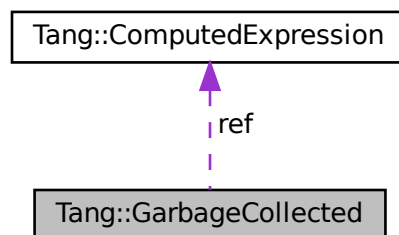
- [include/error.hpp](#)
- [src/error.cpp](#)

5.17 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 [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.

5.17.1 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.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 GarbageCollected() [1/3]

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

Copy Constructor.

Parameters

<i>The</i>	other GarbageCollected object to copy.
------------	--

5.17.2.2 GarbageCollected() [2/3]

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

Move Constructor.

Parameters

<i>The</i>	other GarbageCollected object to move.
------------	--

5.17.2.3 ~GarbageCollected()

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

Destructor.

Clean up the tracked object, if appropriate.

5.17.2.4 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

<i>variable</i>	The arguments to pass to the constructor of the specified type.
-----------------	---

5.17.3 Member Function Documentation**5.17.3.1 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

<i>variable</i>	The arguments to pass to the constructor of the specified type.
-----------------	---

Returns

A [GarbageCollected](#) object.

Here is the call graph for this function:



5.17.3.2 operator%()

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

Perform a modulo between two `GarbageCollected` values.

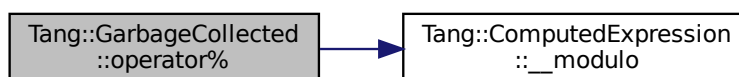
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:



5.17.3.3 operator*() [1/2]

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

Access the tracked object.

Returns

A reference to the tracked object.

5.17.3.4 `operator*()` [2/2]

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

Perform a multiplication between two `GarbageCollected` values.

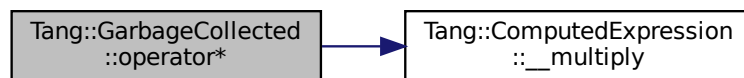
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:



5.17.3.5 `operator+()`

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

Perform an addition between two `GarbageCollected` values.

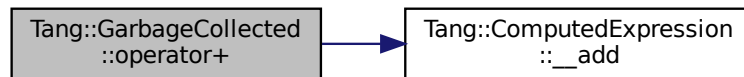
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:

**5.17.3.6 operator-() [1/2]**

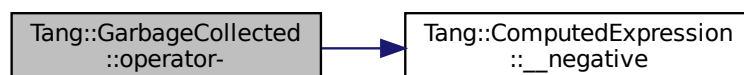
```
GarbageCollected GarbageCollected::operator- ( ) const
```

Perform a negation on the `GarbageCollected` value.

Returns

The result of the operation.

Here is the call graph for this function:

**5.17.3.7 operator-() [2/2]**

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

Perform a subtraction between two `GarbageCollected` values.

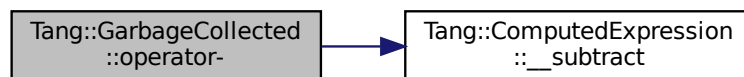
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:

**5.17.3.8 operator->()**

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

Access the tracked object as a pointer.

Returns

A pointer to the tracked object.

5.17.3.9 operator/()

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

Perform a division between two [GarbageCollected](#) values.

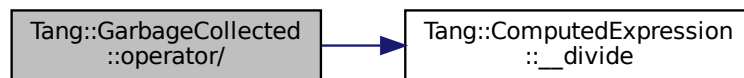
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:



5.17.3.10 operator=() [1/2]

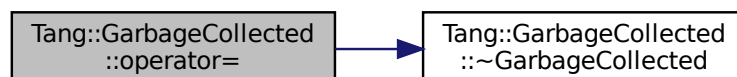
```
GarbageCollected& Tang::GarbageCollected::operator= (
    const GarbageCollected & other ) [inline]
```

Copy Assignment.

Parameters

<i>The</i>	other GarbageCollected object.
------------	--

Here is the call graph for this function:



5.17.3.11 operator=() [2/2]

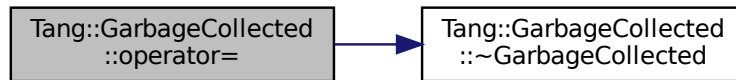
```
GarbageCollected& Tang::GarbageCollected::operator= (
    GarbageCollected && other ) [inline]
```

Move Assignment.

Parameters

<i>The</i>	other GarbageCollected object.
------------	--

Here is the call graph for this function:



5.17.3.12 operator==() [1/3]

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

Compare the [GarbageCollected](#) tracked object with a supplied value.

Parameters

<i>val</i>	The value to compare the tracked object against.
------------	--

Returns

True if they are equal, false otherwise.

5.17.3.13 operator==() [2/3]

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

Compare the [GarbageCollected](#) tracked object with a supplied value.

Parameters

<i>val</i>	The value to compare the tracked object against.
------------	--

Returns

True if they are equal, false otherwise.

5.17.3.14 operator==() [3/3]

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

Compare the [GarbageCollected](#) tracked object with a supplied value.

Parameters

<i>val</i>	The value to compare the tracked object against.
------------	--

Returns

True if they are equal, false otherwise.

5.17.4 Friends And Related Function Documentation**5.17.4.1 operator<<**

```
std::ostream& operator<< (
    std::ostream & out,
    const GarbageCollected & gc ) [friend]
```

Add friendly output.

Parameters

<i>out</i>	The output stream.
<i>gc</i>	The GarbageCollected value.

Returns

The output stream.

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

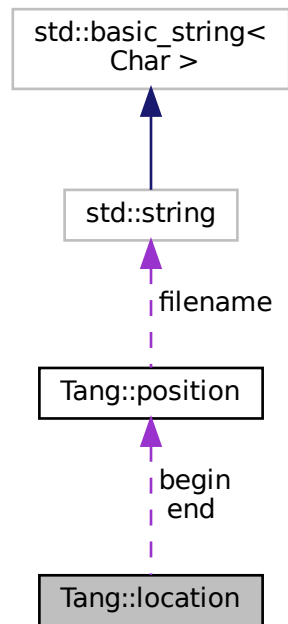
- include/[garbageCollected.hpp](#)
- src/garbageCollected.cpp

5.18 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.

5.18.1 Detailed Description

Two points in a source file.

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

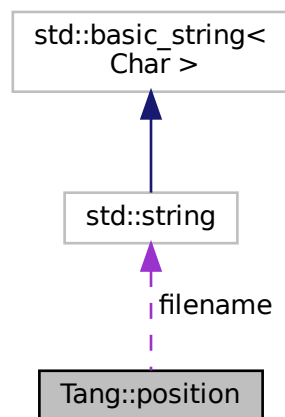
- build/generated/[location.hh](#)

5.19 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.

5.19.1 Detailed Description

A point in a source file.

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

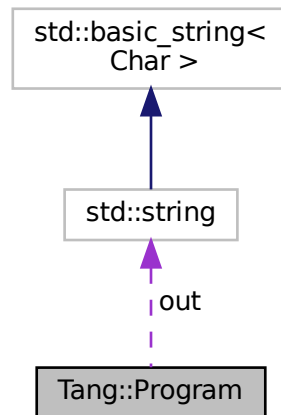
- `build/generated/location.hh`

5.20 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 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.

5.20.1 Detailed Description

Represents a compiled script or template that may be executed.

5.20.2 Member Enumeration Documentation

5.20.2.1 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.

5.20.3 Constructor & Destructor Documentation

5.20.3.1 Program()

```
Program::Program (
    std::string code,
    Program::CodeType codeType )
```

Create a compiled program using the provided code.

Parameters

<i>code</i>	The code to be compiled.
<i>codeType</i>	Whether the code is a <i>Script</i> or <i>Template</i> .

5.20.4 Member Function Documentation

5.20.4.1 addBytecode()

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

Add a `uint64_t` to the Bytecode.

Parameters

<i>op</i>	The value to add to the Bytecode.
-----------	-----------------------------------

5.20.4.2 dumpBytecode()

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

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

Returns

A string containing the Opcode representation.

5.20.4.3 execute()

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

Execute the program's Bytecode, and return the current [Program](#) object.

Returns

The current [Program](#) object.

5.20.4.4 getAst()

```
optional< const 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.

5.20.4.5 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.

5.20.4.6 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](#)

5.21 [Tang::SingletonObjectPool< T >](#) Class Template Reference

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.

5.21.1 Member Function Documentation

5.21.1.1 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.

5.21.1.2 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.

5.21.1.3 recycle()

```
template<class T >
void Tang::SingletonObjectPool< T >::recycle (
    T * obj ) [inline]
```

Recycle a memory location for an object T.

Parameters

<i>obj</i>	The memory location to recycle.
------------	---------------------------------

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

- include/[singletonObjectPool.hpp](#)

5.22 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](#).

5.22.1 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).

5.22.2 Constructor & Destructor Documentation

5.22.2.1 TangBase()

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

The constructor.

Isn't it glorious.

5.22.3 Member Function Documentation

5.22.3.1 compileScript()

```
Program TangBase::compileScript (
    std::string script )
```

Compile the provided source code as a script and return a [Program](#).

Parameters

<i>script</i>	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:

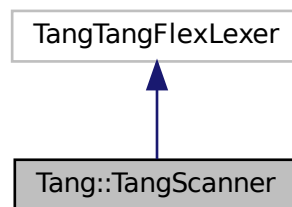
- include/[tangBase.hpp](#)
- src/[tangBase.cpp](#)

5.23 Tang::TangScanner Class Reference

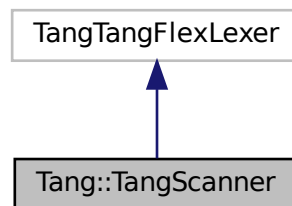
The Flex lexer class for the main Tang language.

```
#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.

5.23.1 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 "`TangTangFlexLexer`". We are subclassing it so that we can override the return type of `get_next_token()`, for compatibility with Bison 3 tokens.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 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

<i>arg_yyin</i>	The input stream to be tokenized
<i>arg_yyout</i>	The output stream (not currently used)

5.23.3 Member Function Documentation

5.23.3.1 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

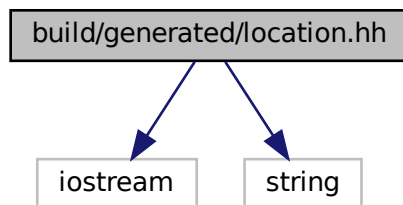
6.1 build/generated/location.hh File Reference

Define the Tang ::location class.

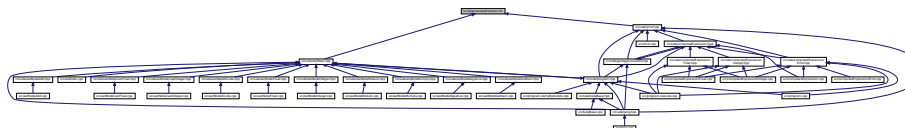
```
#include <iostream>
```

```
#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

<i>ostr</i>	the destination output stream
<i>loc</i>	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.

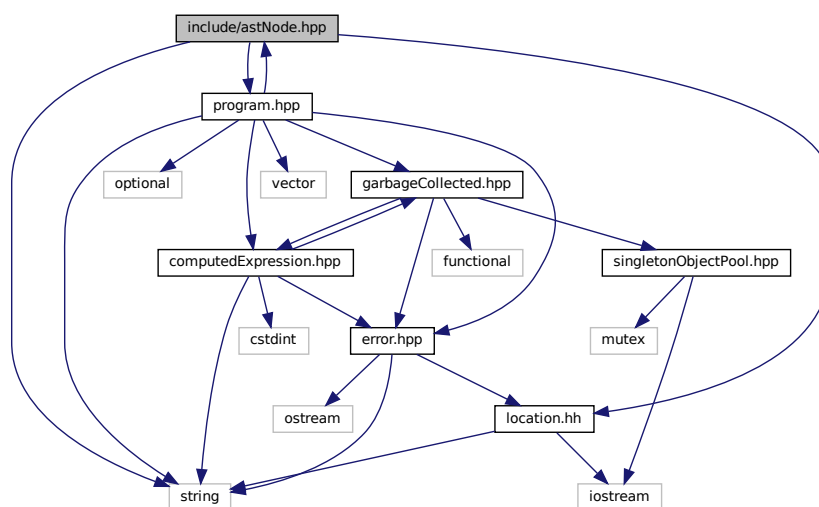
Parameters

<i>ostr</i>	the destination output stream
<i>pos</i>	a reference to the position to redirect

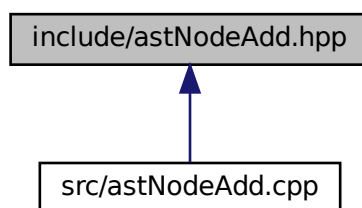
6.2 include/astNode.hpp File Reference

Define the [Tang::AstNode](#) and its associated/derivative classes.

```
#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:



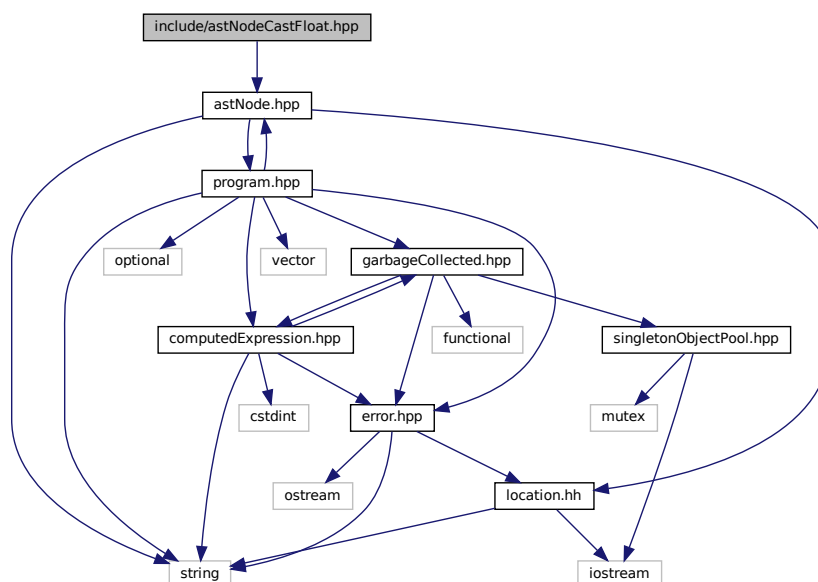
Classes

- class [Tang::AstNodeAdd](#)
An [AstNode](#) that represents a "+" expression.

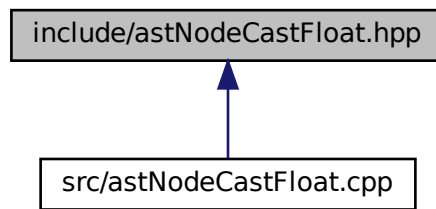
6.4 include/astNodeCastFloat.hpp File Reference

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

Include dependency graph for astNodeCastFloat.hpp:



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



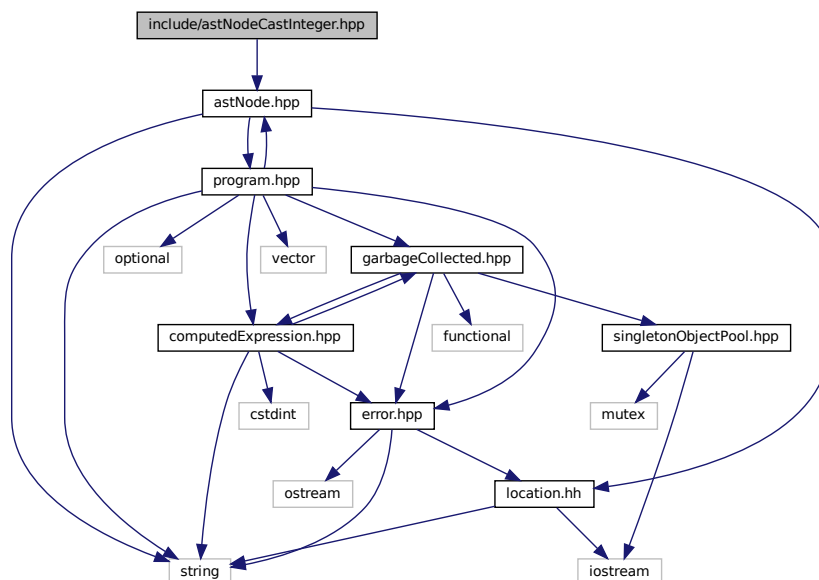
Classes

- class [Tang::AstNodeCastFloat](#)
An [AstNode](#) that represents a typecast to a float.

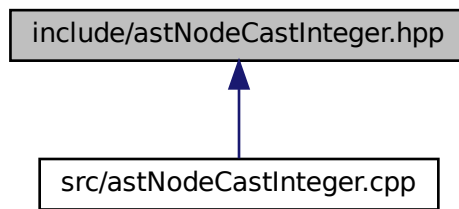
6.5 include/astNodeCastInteger.hpp File Reference

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

Include dependency graph for astNodeCastInteger.hpp:



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



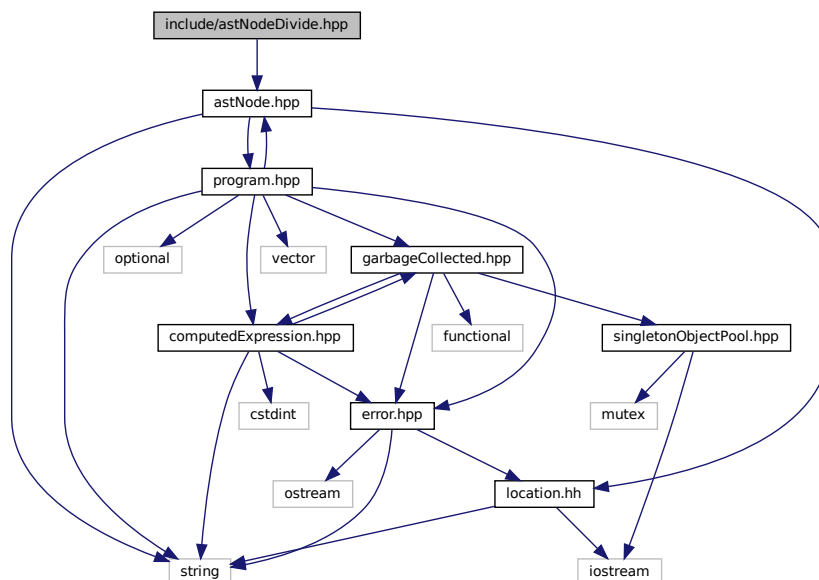
Classes

- class [Tang::AstNodeCastInteger](#)
An [AstNode](#) that represents a typecast to an integer.

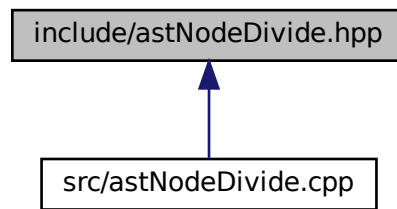
6.6 include/astNodeDivide.hpp File Reference

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

Include dependency graph for astNodeDivide.hpp:



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



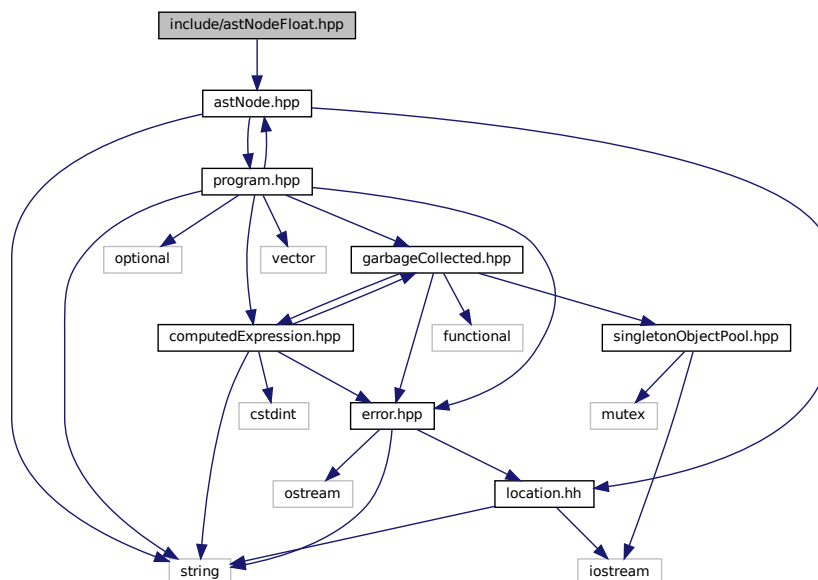
Classes

- class [Tang::AstNodeDivide](#)
An [AstNode](#) that represents a "/" expression.

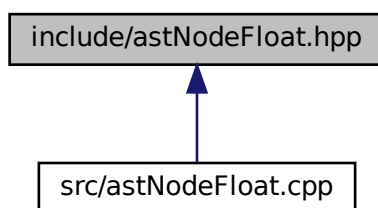
6.7 include/astNodeFloat.hpp File Reference

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

Include dependency graph for astNodeFloat.hpp:



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



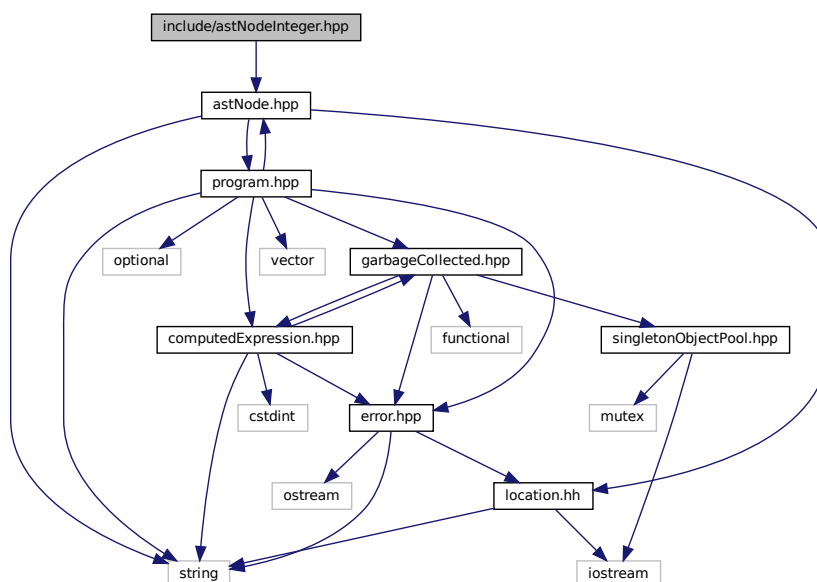
Classes

- class [Tang::AstNodeFloat](#)
An [AstNode](#) that represents an float literal.

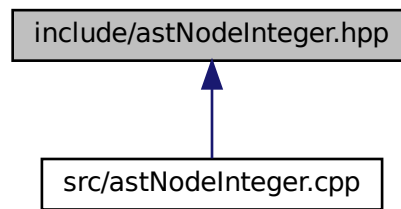
6.8 include/astNodeInteger.hpp File Reference

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

Include dependency graph for astNodeInteger.hpp:



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



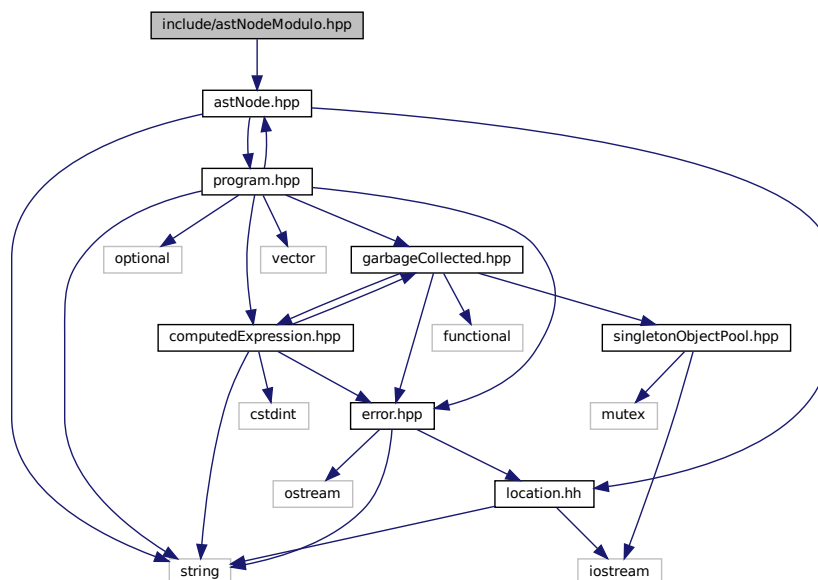
Classes

- class [Tang::AstNodeInteger](#)
An [AstNode](#) that represents an integer literal.

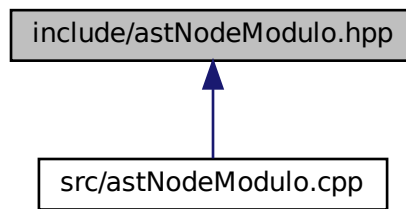
6.9 include/astNodeModulo.hpp File Reference

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

Include dependency graph for astNodeModulo.hpp:



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



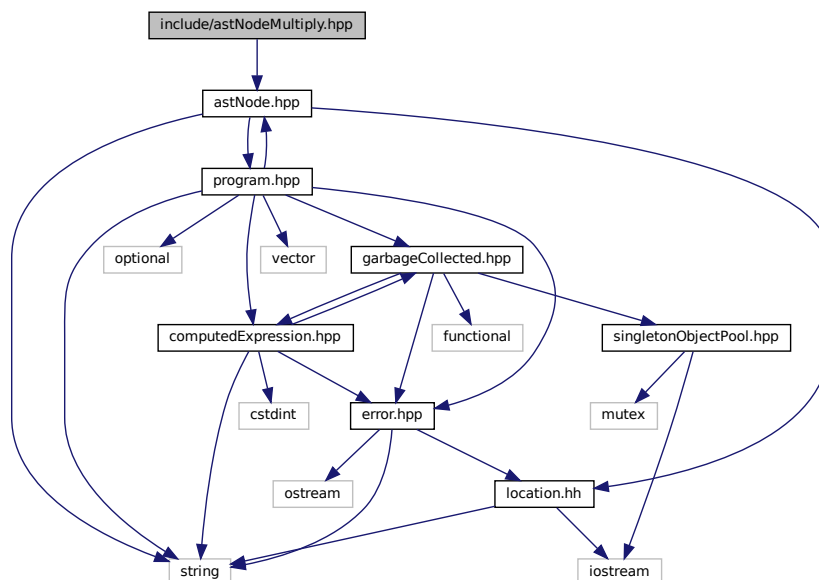
Classes

- class [Tang::AstNodeModulo](#)
An [AstNode](#) that represents a "%" expression.

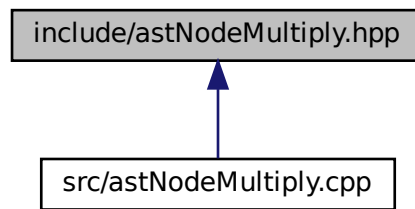
6.10 include/astNodeMultiply.hpp File Reference

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

Include dependency graph for astNodeMultiply.hpp:



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



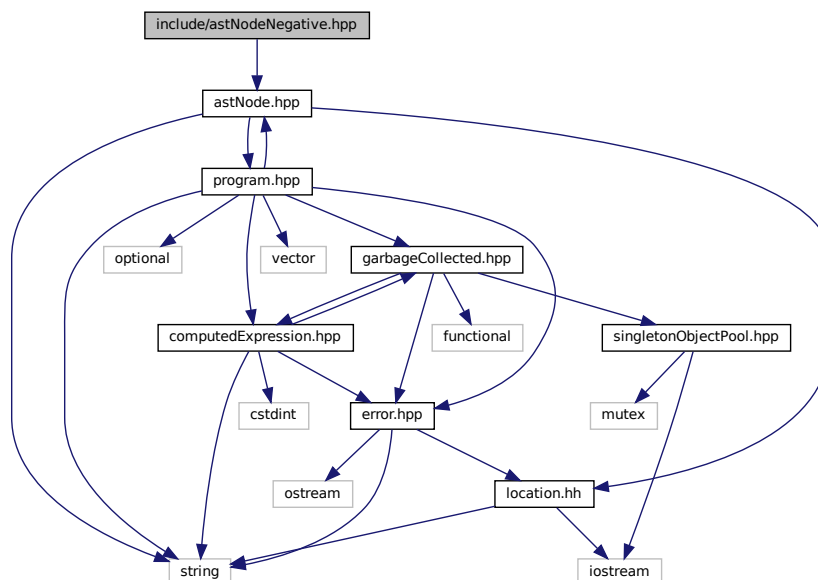
Classes

- class [Tang::AstNodeMultiply](#)
An [AstNode](#) that represents a "*" expression.

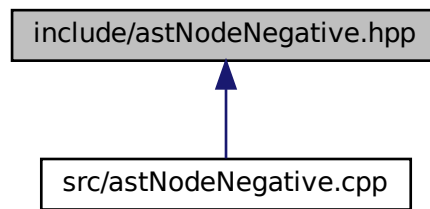
6.11 include/astNodeNegative.hpp File Reference

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

Include dependency graph for astNodeNegative.hpp:



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



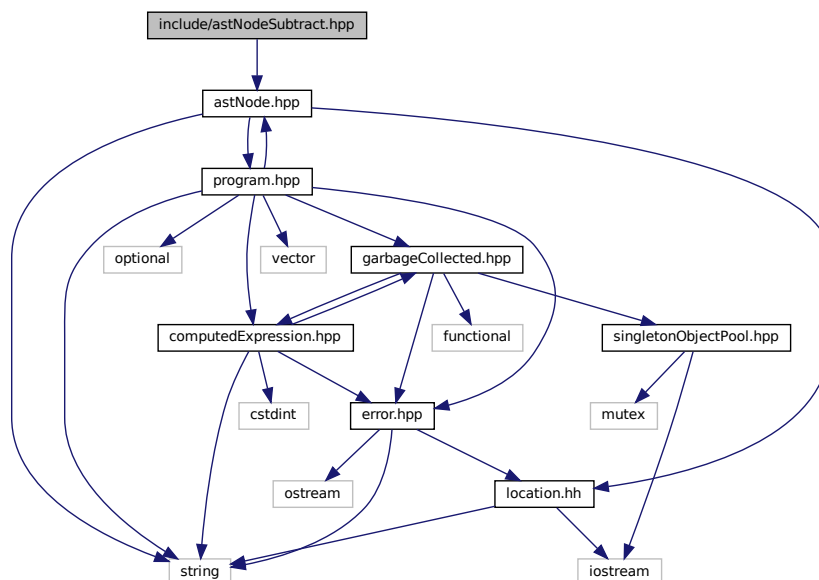
Classes

- class [Tang::AstNodeNegative](#)
An [AstNode](#) that represents a unary negation.

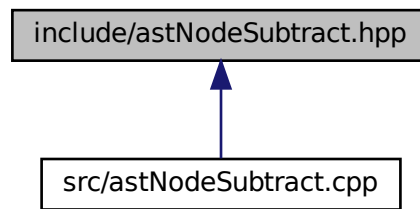
6.12 include/astNodeSubtract.hpp File Reference

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

Include dependency graph for `astNodeSubtract.hpp`:



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

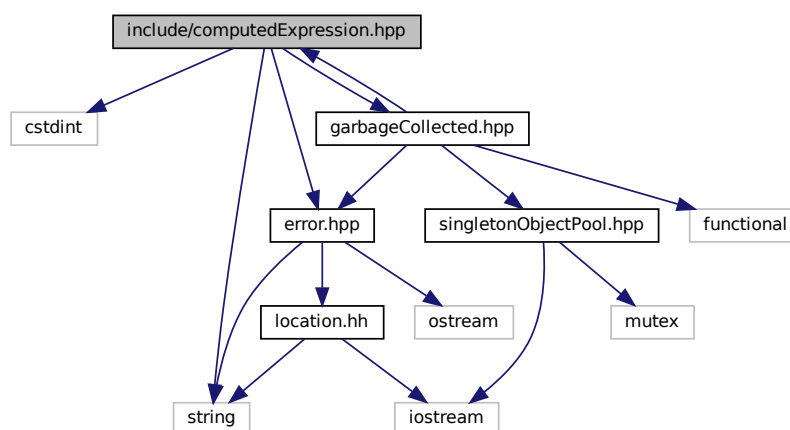


Classes

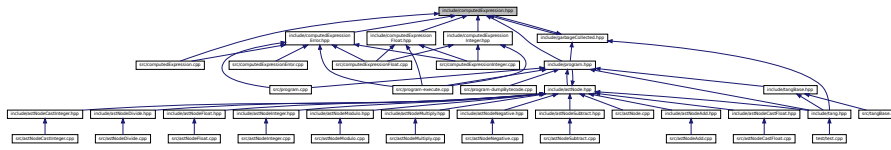
- class [Tang::AstNodeSubtract](#)
An [AstNode](#) that represents a "-" expression.

6.13 include/computedExpression.hpp File Reference

```
#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

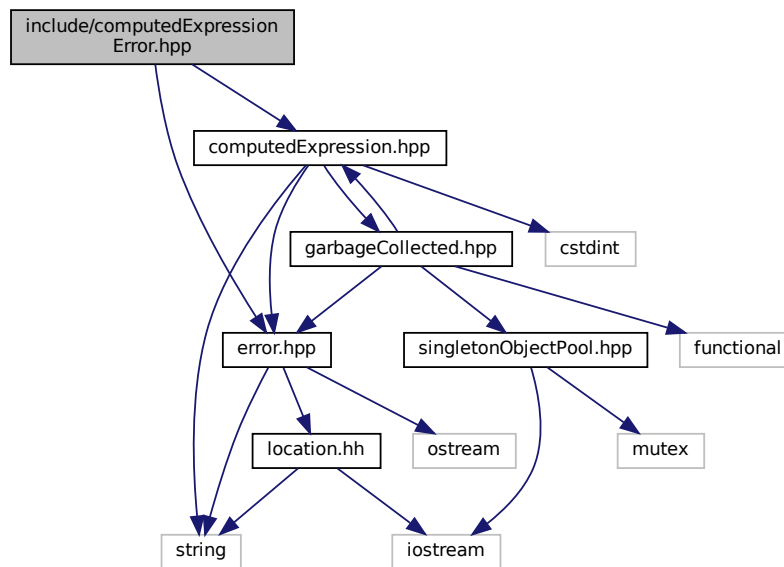
- class [Tang::ComputedExpression](#)

Represents the result of a computation that has been executed.

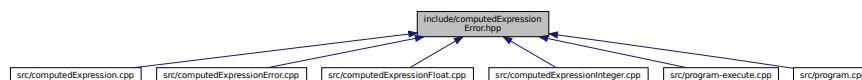
6.14 include/computedExpressionError.hpp File Reference

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

Include dependency graph for computedExpressionError.hpp:



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



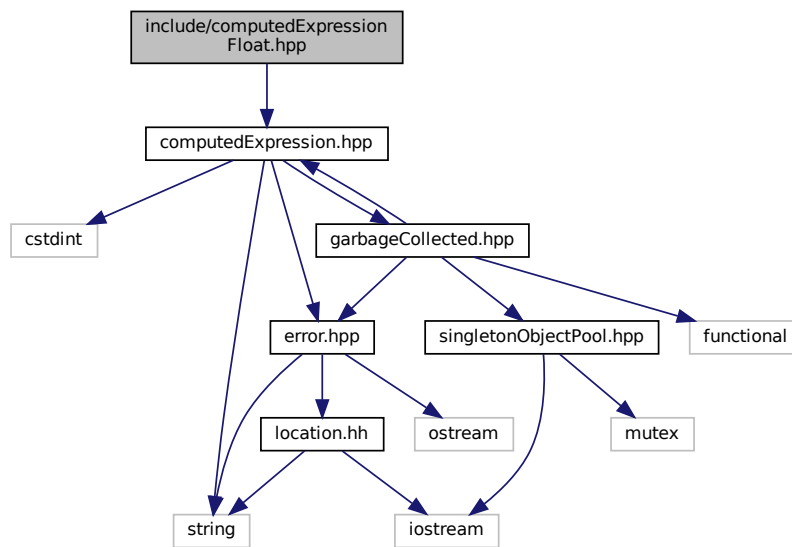
Classes

- class [Tang::ComputedExpressionError](#)
Represents a Runtime [Error](#).

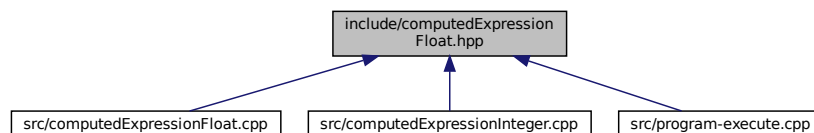
6.15 include/computedExpressionFloat.hpp File Reference

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

Include dependency graph for computedExpressionFloat.hpp:



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

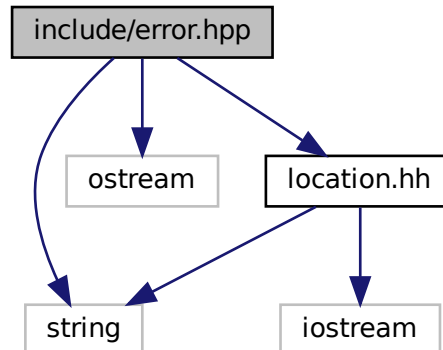


Classes

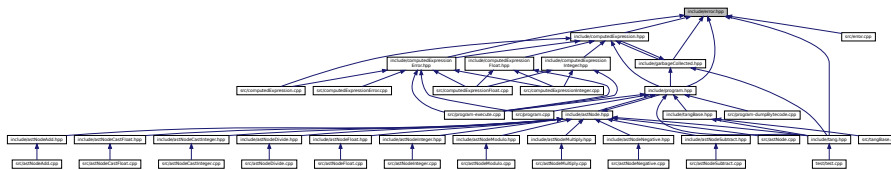
- class [Tang::ComputedExpressionFloat](#)
Represents a Float that is the result of a computation.


```
#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.

6.17.1 Detailed Description

Define the [Tang::Error](#) class used to describe syntax and runtime errors.

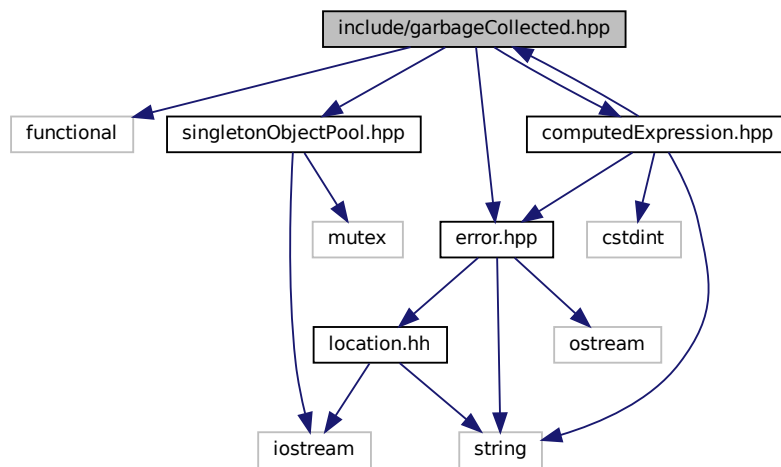
6.18 include/garbageCollected.hpp File Reference

```
#include <functional>
#include "singletonObjectPool.hpp"
#include "computedExpression.hpp"
```

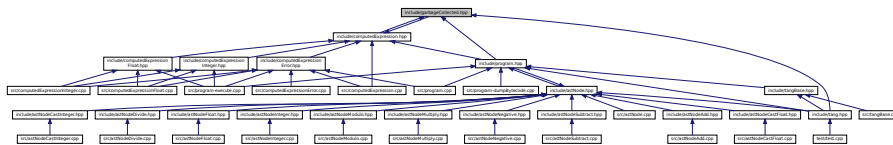


```
#include "error.hpp"
```

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.

6.19 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.

6.19.1 Detailed Description

Contains generic macros.

Macros

- #define [GROW](#) 1024

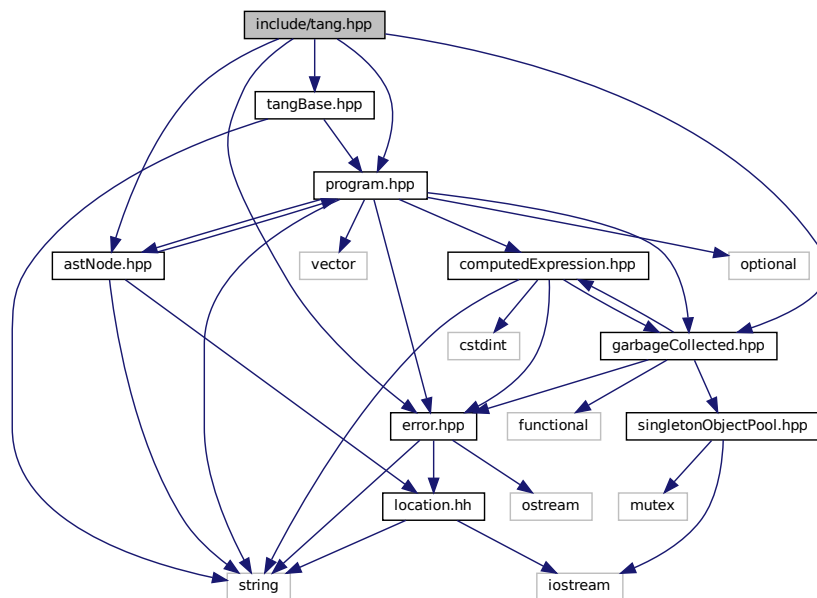
The threshold size to use when allocating blocks of data, measured in the number of instances of the object type.

6.23 include/tang.hpp File Reference

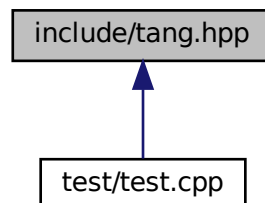
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

```
#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:



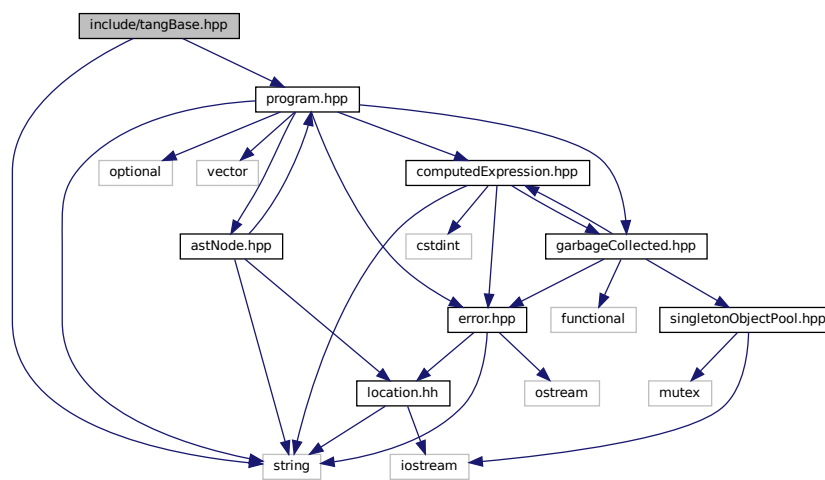
6.23.1 Detailed Description

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

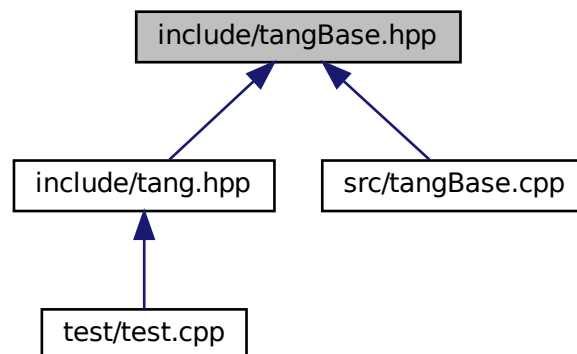
6.24 include/tangBase.hpp File Reference

Defines 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.

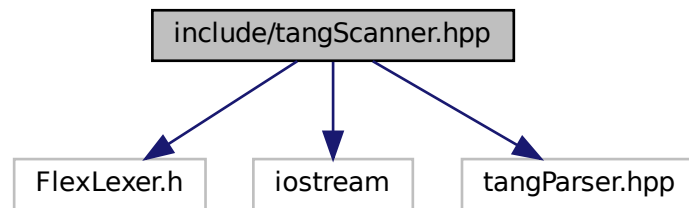
6.24.1 Detailed Description

Defines the [Tang::TangBase](#) class used to interact with Tang.

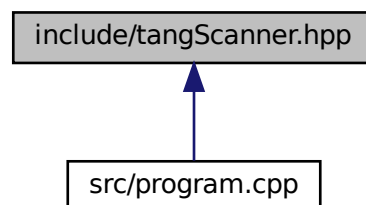
6.25 include/tangScanner.hpp File Reference

Defines 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\(\)](#)

6.25.1 Detailed Description

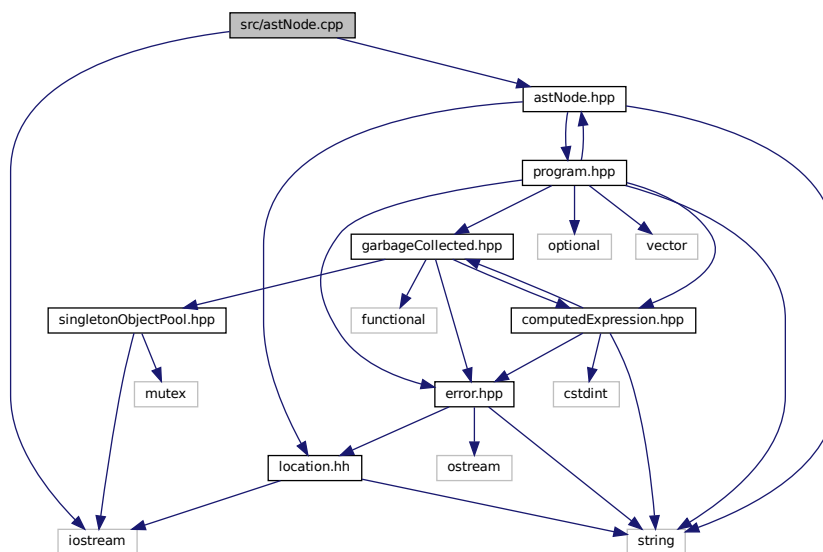
Defines the [Tang::TangScanner](#) used to tokenize a Tang script.

6.26 src/astNode.cpp File Reference

```
#include <iostream>
```

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

Include dependency graph for astNode.cpp:

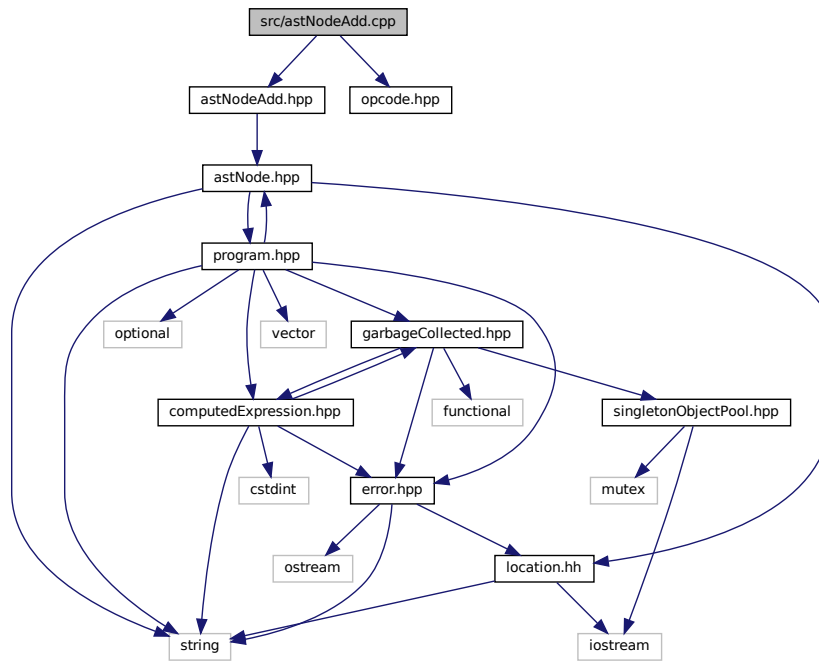


6.27 src/astNodeAdd.cpp File Reference

```
#include "astNodeAdd.hpp"
```

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

Include dependency graph for astNodeAdd.cpp:



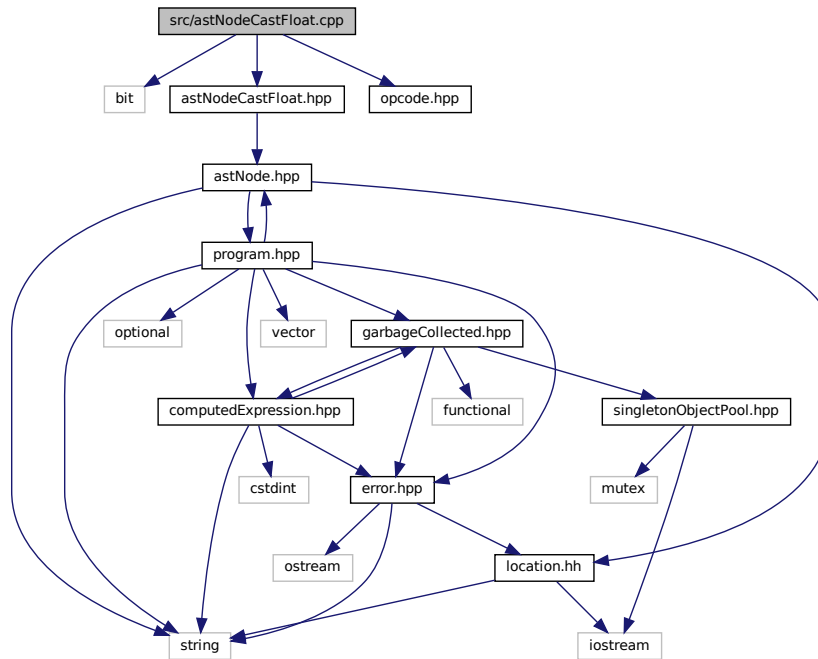
6.28 src/astNodeCastFloat.cpp File Reference

```
#include <bit>
```

```
#include "astNodeCastFloat.hpp"
```

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

Include dependency graph for astNodeCastFloat.cpp:



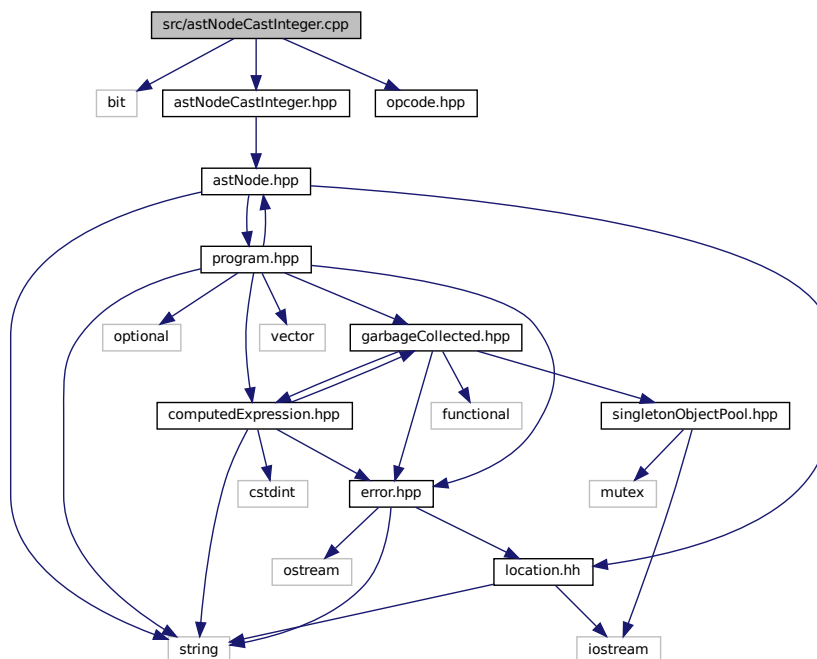
6.29 src/astNodeCastInteger.cpp File Reference

```

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

```

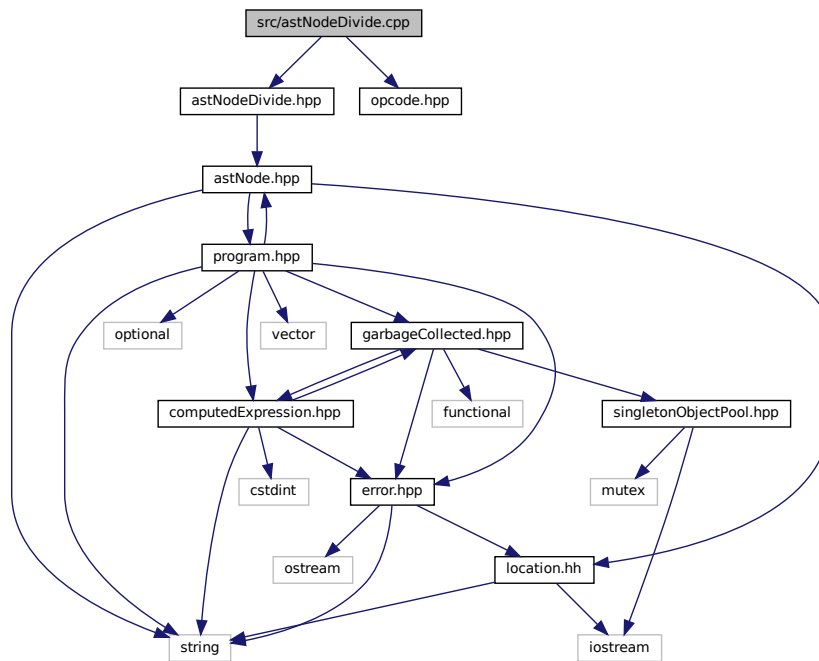
Include dependency graph for astNodeCastInteger.cpp:



6.30 src/astNodeDivide.cpp File Reference

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

Include dependency graph for astNodeDivide.cpp:



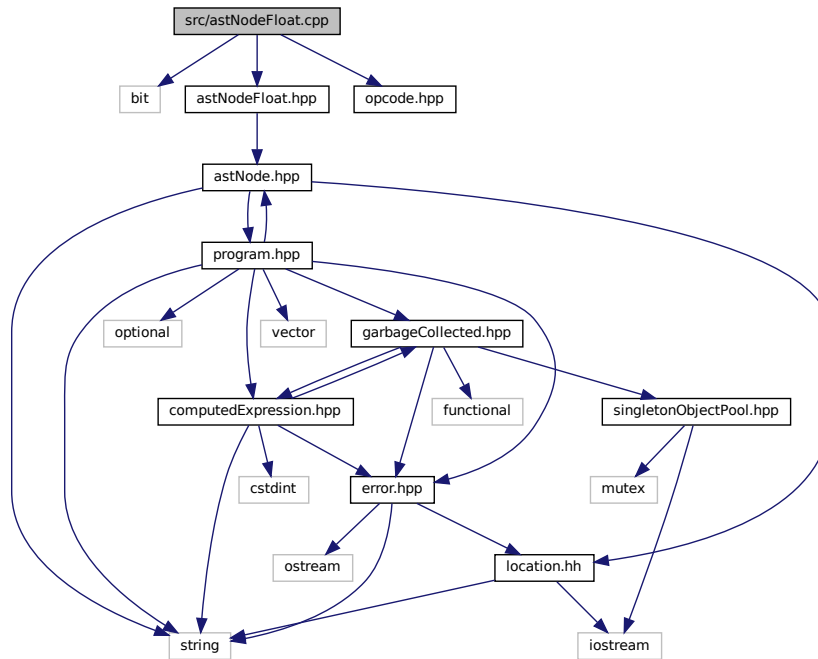
6.31 src/astNodeFloat.cpp File Reference

```

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

```

Include dependency graph for astNodeFloat.cpp:



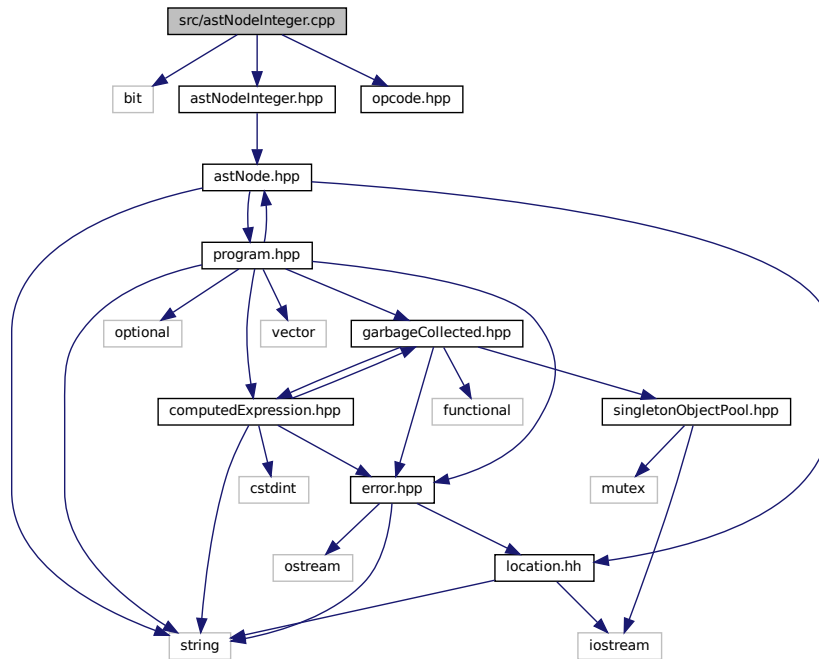
6.32 src/astNodeInteger.cpp File Reference

```

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

```

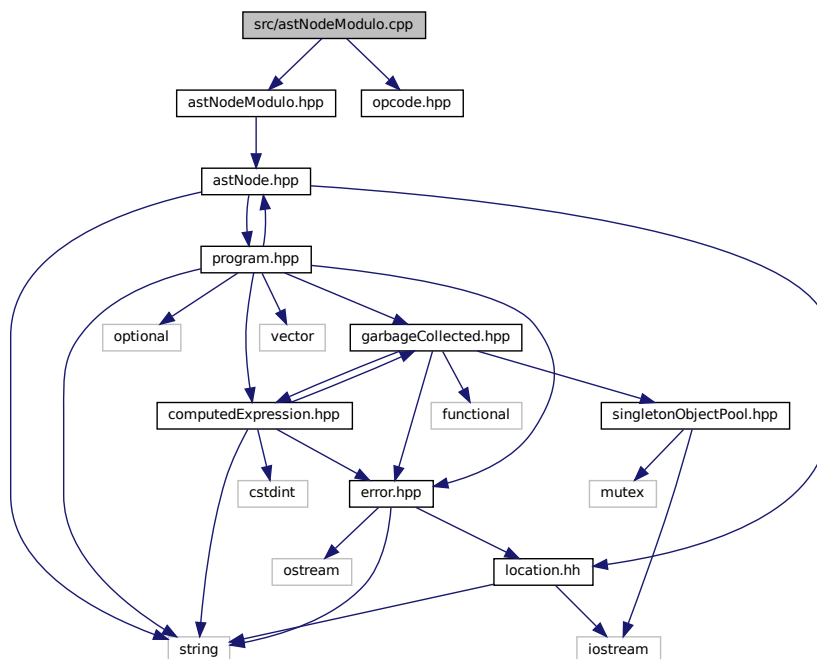
Include dependency graph for astNodeInteger.cpp:



6.33 src/astNodeModulo.cpp File Reference

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

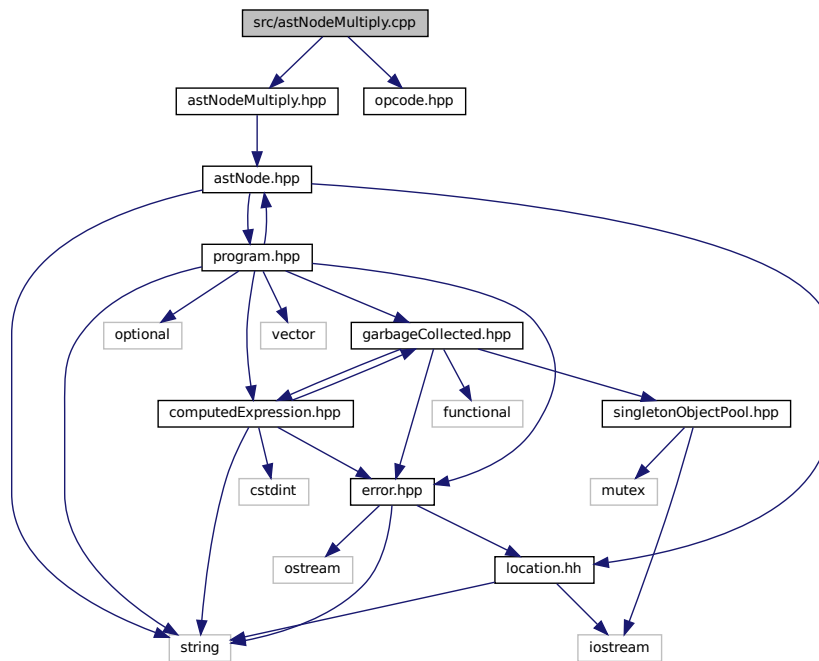
Include dependency graph for astNodeModulo.cpp:



6.34 src/astNodeMultiply.cpp File Reference

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

Include dependency graph for astNodeMultiply.cpp:



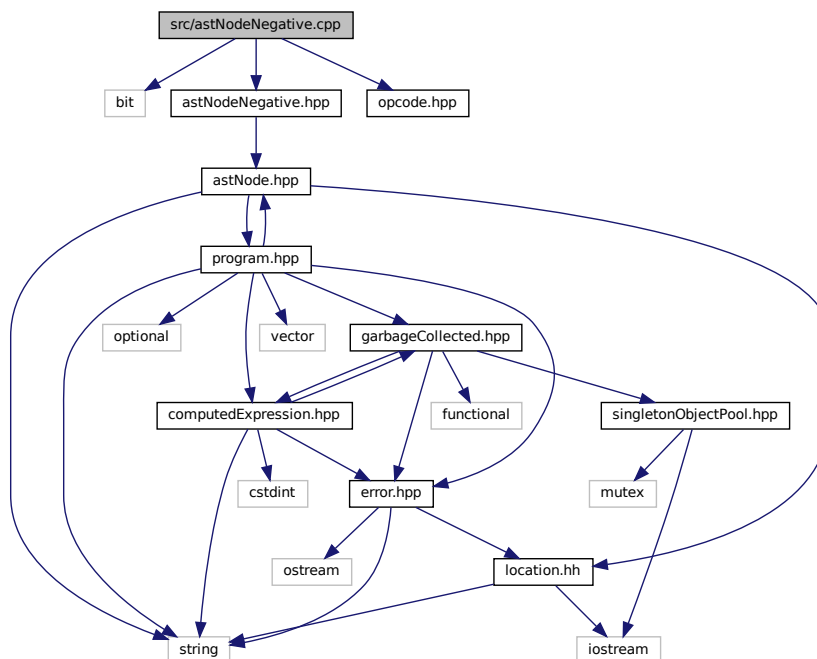
6.35 src/astNodeNegative.cpp File Reference

```

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

```

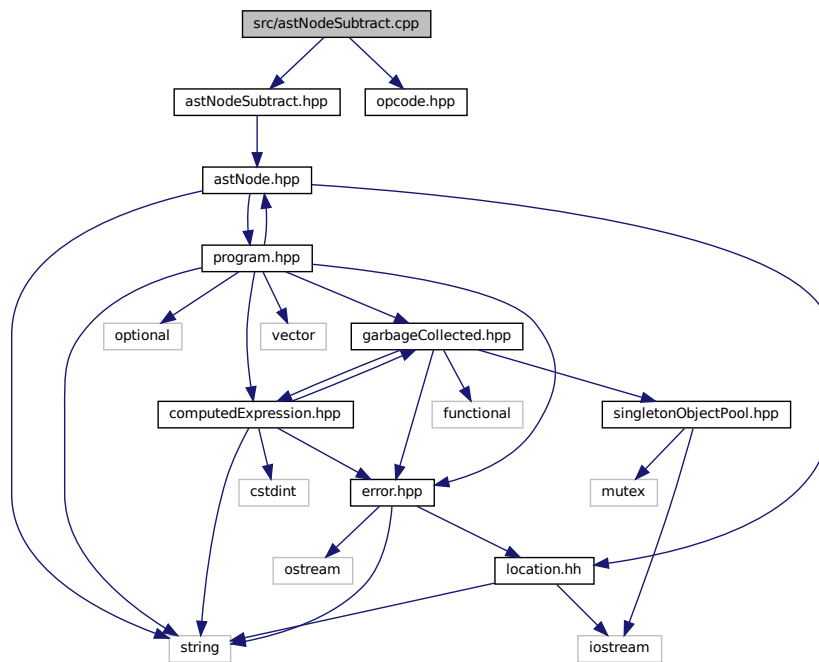

Include dependency graph for astNodeNegative.cpp:



6.36 src/astNodeSubtract.cpp File Reference

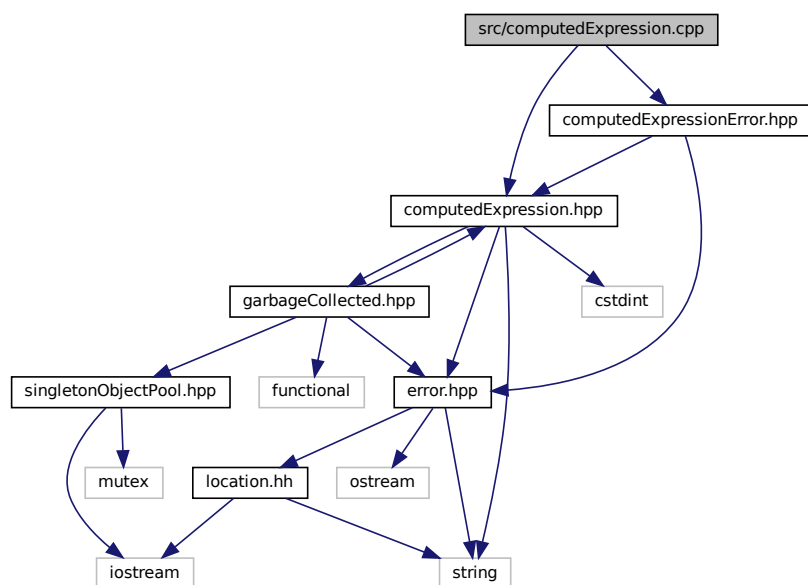
```
#include "astNodeSubtract.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeSubtract.cpp:



6.37 src/computedExpression.cpp File Reference

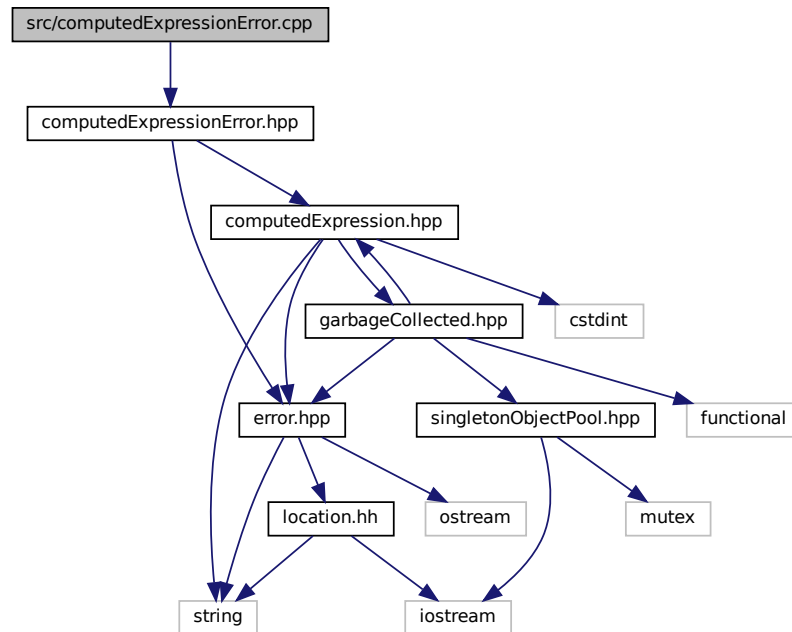
```
#include "computedExpression.hpp"
#include "computedExpressionError.hpp"
Include dependency graph for computedExpression.cpp:
```



6.38 src/computedExpressionError.cpp File Reference

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

Include dependency graph for computedExpressionError.cpp:



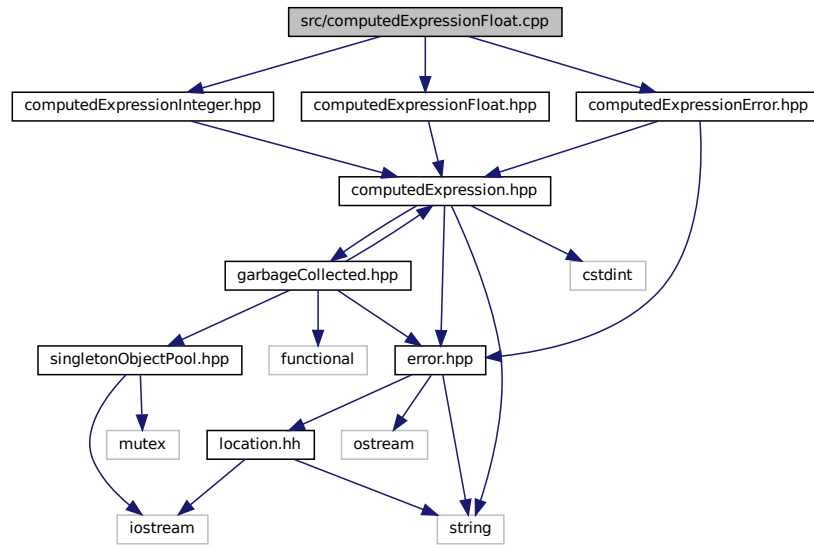
6.39 src/computedExpressionFloat.cpp File Reference

```
#include "computedExpressionFloat.hpp"
```

```
#include "computedExpressionInteger.hpp"
```

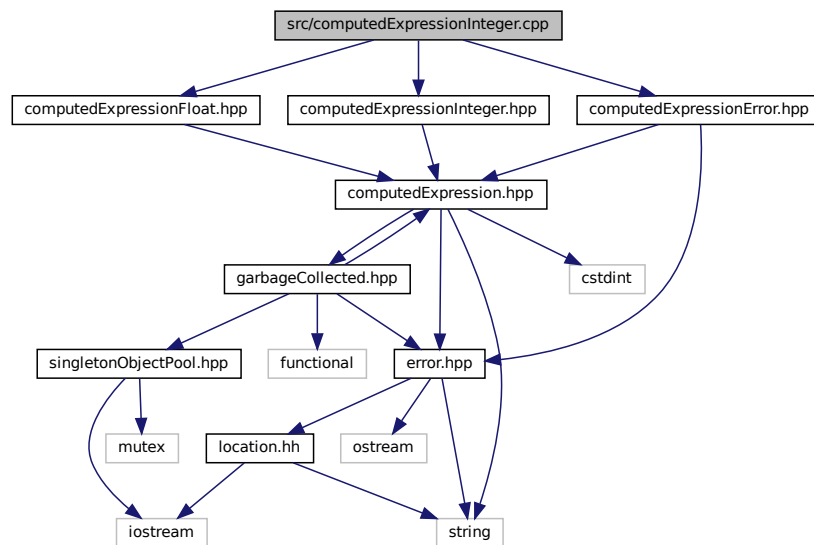
```
#include "computedExpressionError.hpp"
```

Include dependency graph for `computedExpressionFloat.cpp`:



6.40 `src/computedExpressionInteger.cpp` File Reference

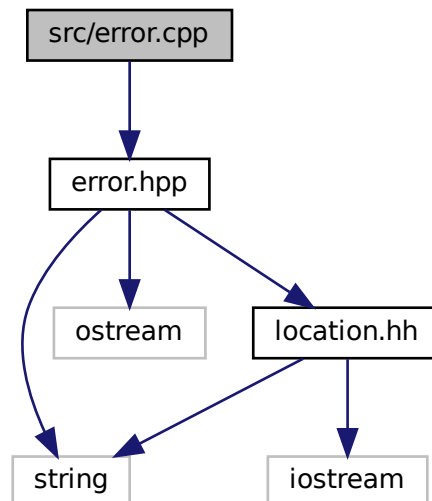
```
#include "computedExpressionInteger.hpp"
#include "computedExpressionFloat.hpp"
#include "computedExpressionError.hpp"
Include dependency graph for computedExpressionInteger.cpp:
```



6.41 src/error.cpp File Reference

```
#include "error.hpp"
```

Include dependency graph for error.cpp:



Functions

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

6.41.1 Function Documentation

6.41.1.1 operator<<()

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

Parameters

<i>out</i>	The output stream.
<i>error</i>	The Error object.

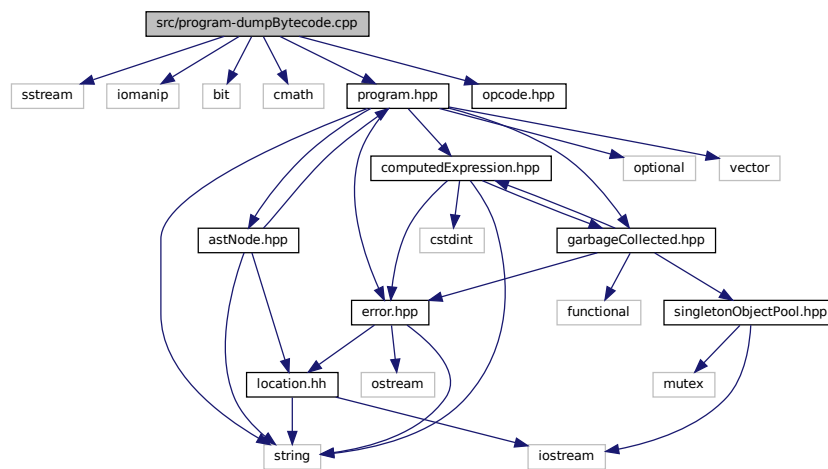
Returns

The output stream.

6.42 src/program-dumpBytecode.cpp File Reference

```
#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.

6.42.1 Macro Definition Documentation

6.42.1.1 DUMPPROGRAMCHECK

```
#define DUMPPROGRAMCHECK(  
    x )
```

Value:

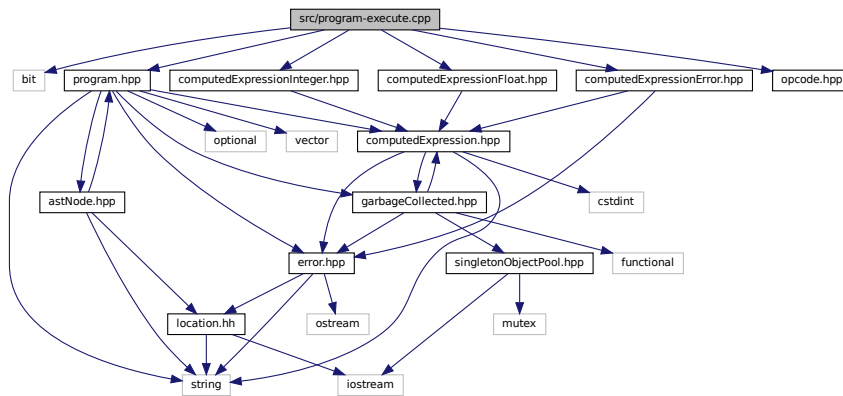
```
if (this->bytecode.size() < (pc + (x))) \
    return out.str() + "Error: Opcode truncated\n"
```

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.

x	The number of additional vector entries that should exist.
-----	--

```
#include <bit>
#include "program.hpp"
#include "opcode.hpp"
#include "computedExpressionError.hpp"
#include "computedExpressionInteger.hpp"
#include "computedExpressionFloat.hpp"
Include dependency graph for program-execute.cpp:
```



- #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.

6.43.1.1 EXECUTEPROGRAMCHECK

```

if (this->bytecode.size() < (pc + (x))) { \
    stack.push_back(GarbageCollected::make<ComputedExpressionError>(Error{"Opcode instruction\n\
        truncated."})); \
    pc = this->bytecode.size(); \
    break; \
}

```

Generated by Doxygen

Parameters

x	The number of additional vector entries that should exist.
---	--

6.43.1.2 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; \
}
```

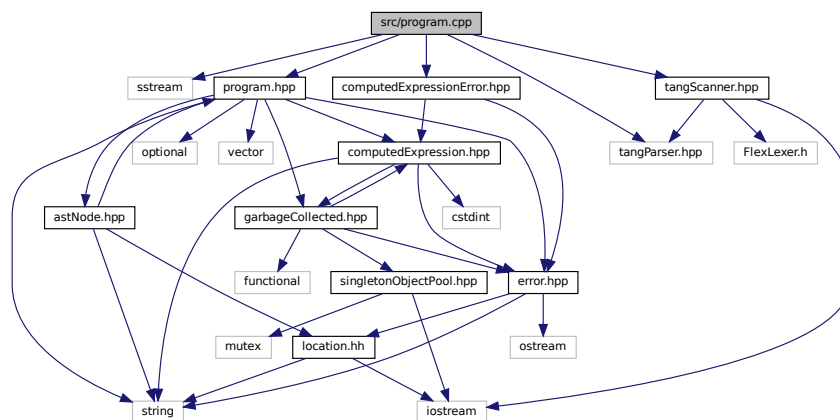
Verify the size of the stack vector so that it may be safely accessed.

Parameters

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

6.44 src/program.cpp File Reference

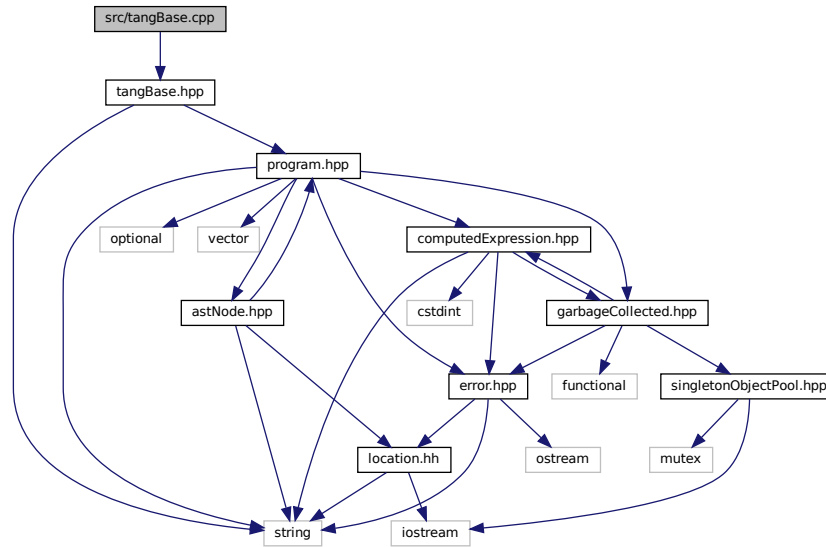
```
#include <sstream>
#include "program.hpp"
#include "tangScanner.hpp"
#include "tangParser.hpp"
#include "computedExpressionError.hpp"
Include dependency graph for program.cpp:
```



6.45 src/tangBase.cpp File Reference

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

Include dependency graph for tangBase.cpp:



6.46 test/test.cpp File Reference

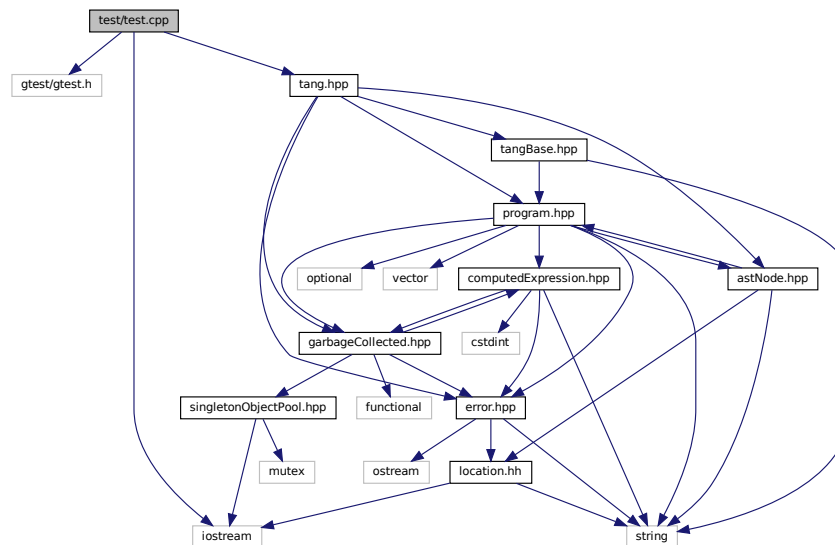
Test the general language behaviors.

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

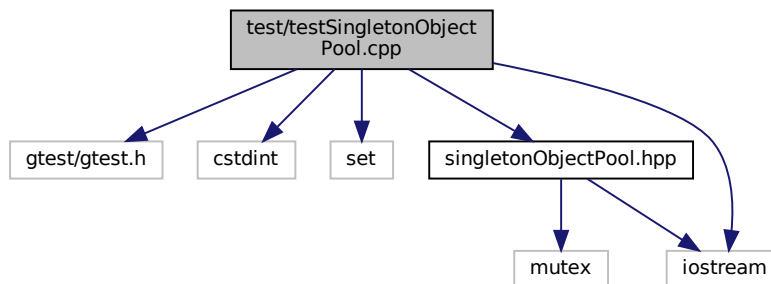
6.46.1 Detailed Description

Test the general language behaviors.

6.47 test/testSingletonObjectPool.cpp File Reference

```
#include <gtest/gtest.h>
#include <stdint>
#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)

Index

- __add
 - Tang::ComputedExpression, [44](#)
 - Tang::ComputedExpressionError, [50](#)
 - Tang::ComputedExpressionFloat, [57](#)
 - Tang::ComputedExpressionInteger, [64](#)
 - __divide
 - Tang::ComputedExpression, [44](#)
 - Tang::ComputedExpressionError, [51](#)
 - Tang::ComputedExpressionFloat, [57](#)
 - Tang::ComputedExpressionInteger, [64](#)
 - __float
 - Tang::ComputedExpression, [44](#)
 - Tang::ComputedExpressionError, [51](#)
 - Tang::ComputedExpressionFloat, [58](#)
 - Tang::ComputedExpressionInteger, [64](#)
 - __integer
 - Tang::ComputedExpression, [45](#)
 - Tang::ComputedExpressionError, [51](#)
 - Tang::ComputedExpressionFloat, [58](#)
 - Tang::ComputedExpressionInteger, [65](#)
 - __modulo
 - Tang::ComputedExpression, [45](#)
 - Tang::ComputedExpressionError, [52](#)
 - Tang::ComputedExpressionFloat, [58](#)
 - Tang::ComputedExpressionInteger, [65](#)
 - __multiply
 - Tang::ComputedExpression, [45](#)
 - Tang::ComputedExpressionError, [52](#)
 - Tang::ComputedExpressionFloat, [59](#)
 - Tang::ComputedExpressionInteger, [65](#)
 - __negative
 - Tang::ComputedExpression, [46](#)
 - Tang::ComputedExpressionError, [52](#)
 - Tang::ComputedExpressionFloat, [59](#)
 - Tang::ComputedExpressionInteger, [66](#)
 - __subtract
 - Tang::ComputedExpression, [46](#)
 - Tang::ComputedExpressionError, [53](#)
 - Tang::ComputedExpressionFloat, [59](#)
 - Tang::ComputedExpressionInteger, [66](#)
- ~GarbageCollected
 - Tang::GarbageCollected, [74](#)
- ADD
 - opcode.hpp, [113](#)
- addBytecode
 - Tang::Program, [86](#)
- AstNode
 - Tang::AstNode, [12](#)
- AstNodeAdd
 - Tang::AstNodeAdd, [15](#)
- AstNodeCastFloat
 - Tang::AstNodeCastFloat, [18](#)
- AstNodeCastInteger
 - Tang::AstNodeCastInteger, [21](#)
- AstNodeDivide
 - Tang::AstNodeDivide, [24](#)
- AstNodeFloat
 - Tang::AstNodeFloat, [27](#)
- AstNodeInteger
 - Tang::AstNodeInteger, [30](#)
- AstNodeModulo
 - Tang::AstNodeModulo, [33](#)
- AstNodeMultiply
 - Tang::AstNodeMultiply, [36](#)
- AstNodeNegative
 - Tang::AstNodeNegative, [39](#)
- AstNodeSubtract
 - Tang::AstNodeSubtract, [42](#)
- build/generated/location.hh, [93](#)
- CASTFLOAT
 - opcode.hpp, [113](#)
- CASTINTEGER
 - opcode.hpp, [113](#)
- CodeType
 - Tang::Program, [86](#)
- compileScript
 - Tang::TangBase, [90](#)
- ComputedExpressionError
 - Tang::ComputedExpressionError, [50](#)
- ComputedExpressionFloat
 - Tang::ComputedExpressionFloat, [57](#)
- ComputedExpressionInteger
 - Tang::ComputedExpressionInteger, [63](#)
- DIVIDE
 - opcode.hpp, [113](#)
- dump
 - Tang::ComputedExpression, [46](#)
 - Tang::ComputedExpressionError, [53](#)
 - Tang::ComputedExpressionFloat, [60](#)
 - Tang::ComputedExpressionInteger, [66](#)
- dumpBytecode
 - Tang::Program, [87](#)
- DUMPPROGRAMCHECK
 - program-dumpBytecode.cpp, [132](#)
- Error

- Tang::Error, 70
- error.cpp
 - operator<<, 131
- execute
 - Tang::Program, 87
- EXECUTEPROGRAMCHECK
 - program-execute.cpp, 133
- FLOAT
 - opcode.hpp, 113
- GarbageCollected
 - Tang::GarbageCollected, 73, 74
- get
 - Tang::SingletonObjectPool< T >, 88
- get_next_token
 - Tang::TangScanner, 92
- getAst
 - Tang::Program, 87
- getCode
 - Tang::Program, 87
- getInstance
 - Tang::SingletonObjectPool< T >, 89
- getResult
 - Tang::Program, 88
- include/astNode.hpp, 95
- include/astNodeAdd.hpp, 96
- include/astNodeCastFloat.hpp, 97
- include/astNodeCastInteger.hpp, 98
- include/astNodeDivide.hpp, 99
- include/astNodeFloat.hpp, 100
- include/astNodeInteger.hpp, 101
- include/astNodeModulo.hpp, 102
- include/astNodeMultiply.hpp, 103
- include/astNodeNegative.hpp, 104
- include/astNodeSubtract.hpp, 105
- include/computedExpression.hpp, 106
- include/computedExpressionError.hpp, 107
- include/computedExpressionFloat.hpp, 108
- include/computedExpressionInteger.hpp, 109
- include/error.hpp, 109
- include/garbageCollected.hpp, 110
- include/macros.hpp, 111
- include/opcode.hpp, 112
- include/program.hpp, 113
- include/singletonObjectPool.hpp, 114
- include/tang.hpp, 115
- include/tangBase.hpp, 116
- include/tangScanner.hpp, 117
- INTEGER
 - opcode.hpp, 113
- is_equal
 - Tang::ComputedExpression, 47
 - Tang::ComputedExpressionError, 53, 54
 - Tang::ComputedExpressionFloat, 60, 61
 - Tang::ComputedExpressionInteger, 67
- location.hh
 - operator<<, 94, 95
- macros.hpp
 - TANG_UNUSED, 112
- make
 - Tang::GarbageCollected, 74
- makeCopy
 - Tang::AstNode, 12
 - Tang::AstNodeAdd, 15
 - Tang::AstNodeCastFloat, 18
 - Tang::AstNodeCastInteger, 21
 - Tang::AstNodeDivide, 24
 - Tang::AstNodeFloat, 27
 - Tang::AstNodeInteger, 30
 - Tang::AstNodeModulo, 33
 - Tang::AstNodeMultiply, 36
 - Tang::AstNodeNegative, 39
 - Tang::AstNodeSubtract, 42
 - Tang::ComputedExpression, 48
 - Tang::ComputedExpressionError, 54
 - Tang::ComputedExpressionFloat, 61
 - Tang::ComputedExpressionInteger, 68
- MODULO
 - opcode.hpp, 113
- MULTIPLY
 - opcode.hpp, 113
- NEGATIVE
 - opcode.hpp, 113
- Opcode
 - opcode.hpp, 112
- opcode.hpp
 - ADD, 113
 - CASTFLOAT, 113
 - CASTINTEGER, 113
 - DIVIDE, 113
 - FLOAT, 113
 - INTEGER, 113
 - MODULO, 113
 - MULTIPLY, 113
 - NEGATIVE, 113
 - Opcode, 112
 - SUBTRACT, 113
- operator<<
 - error.cpp, 131
 - location.hh, 94, 95
 - Tang::Error, 70
 - Tang::GarbageCollected, 81
- operator*
 - Tang::GarbageCollected, 75
- operator+
 - Tang::GarbageCollected, 76
- operator-
 - Tang::GarbageCollected, 77
- operator->
 - Tang::GarbageCollected, 78
- operator/
 - Tang::GarbageCollected, 78

- operator=
 - Tang::GarbageCollected, 79
- operator==
 - Tang::GarbageCollected, 80
- operator%
 - Tang::GarbageCollected, 75
- Program
 - Tang::Program, 86
- program-dumpBytecode.cpp
 - DUMPPROGRAMCHECK, 132
- program-execute.cpp
 - EXECUTEPROGRAMCHECK, 133
 - STACKCHECK, 134
- recycle
 - Tang::SingletonObjectPool< T >, 89
- Script
 - Tang::Program, 86
- src/astNode.cpp, 118
- src/astNodeAdd.cpp, 119
- src/astNodeCastFloat.cpp, 119
- src/astNodeCastInteger.cpp, 120
- src/astNodeDivide.cpp, 121
- src/astNodeFloat.cpp, 122
- src/astNodeInteger.cpp, 123
- src/astNodeModulo.cpp, 124
- src/astNodeMultiply.cpp, 125
- src/astNodeNegative.cpp, 126
- src/astNodeSubtract.cpp, 127
- src/computedExpression.cpp, 128
- src/computedExpressionError.cpp, 129
- src/computedExpressionFloat.cpp, 129
- src/computedExpressionInteger.cpp, 130
- src/error.cpp, 131
- src/program-dumpBytecode.cpp, 132
- src/program-execute.cpp, 133
- src/program.cpp, 134
- src/tangBase.cpp, 135
- STACKCHECK
 - program-execute.cpp, 134
- SUBTRACT
 - opcode.hpp, 113
- Tang::AstNode, 9
 - AstNode, 12
 - makeCopy, 12
- Tang::AstNodeAdd, 13
 - AstNodeAdd, 15
 - makeCopy, 15
- Tang::AstNodeCastFloat, 16
 - AstNodeCastFloat, 18
 - makeCopy, 18
- Tang::AstNodeCastInteger, 19
 - AstNodeCastInteger, 21
 - makeCopy, 21
- Tang::AstNodeDivide, 22
 - AstNodeDivide, 24
 - makeCopy, 24
- Tang::AstNodeFloat, 25
 - AstNodeFloat, 27
 - makeCopy, 27
- Tang::AstNodeInteger, 28
 - AstNodeInteger, 30
 - makeCopy, 30
- Tang::AstNodeModulo, 31
 - AstNodeModulo, 33
 - makeCopy, 33
- Tang::AstNodeMultiply, 34
 - AstNodeMultiply, 36
 - makeCopy, 36
- Tang::AstNodeNegative, 37
 - AstNodeNegative, 39
 - makeCopy, 39
- Tang::AstNodeSubtract, 40
 - AstNodeSubtract, 42
 - makeCopy, 42
- Tang::ComputedExpression, 43
 - __add, 44
 - __divide, 44
 - __float, 44
 - __integer, 45
 - __modulo, 45
 - __multiply, 45
 - __negative, 46
 - __subtract, 46
 - dump, 46
 - is_equal, 47
 - makeCopy, 48
- Tang::ComputedExpressionError, 48
 - __add, 50
 - __divide, 51
 - __float, 51
 - __integer, 51
 - __modulo, 52
 - __multiply, 52
 - __negative, 52
 - __subtract, 53
 - ComputedExpressionError, 50
 - dump, 53
 - is_equal, 53, 54
 - makeCopy, 54
- Tang::ComputedExpressionFloat, 55
 - __add, 57
 - __divide, 57
 - __float, 58
 - __integer, 58
 - __modulo, 58
 - __multiply, 59
 - __negative, 59
 - __subtract, 59
 - ComputedExpressionFloat, 57
 - dump, 60
 - is_equal, 60, 61
 - makeCopy, 61
- Tang::ComputedExpressionInteger, 62

- __add, 64
- __divide, 64
- __float, 64
- __integer, 65
- __modulo, 65
- __multiply, 65
- __negative, 66
- __subtract, 66
- ComputedExpressionInteger, 63
- dump, 66
- is_equal, 67
- makeCopy, 68
- Tang::Error, 69
 - Error, 70
 - operator<<, 70
- Tang::GarbageCollected, 71
 - ~GarbageCollected, 74
 - GarbageCollected, 73, 74
 - make, 74
 - operator<<, 81
 - operator*, 75
 - operator+, 76
 - operator-, 77
 - operator->, 78
 - operator/, 78
 - operator=, 79
 - operator==, 80
 - operator%, 75
- Tang::location, 81
- Tang::position, 83
- Tang::Program, 84
 - addBytecode, 86
 - CodeType, 86
 - dumpBytecode, 87
 - execute, 87
 - getAst, 87
 - getCode, 87
 - getResult, 88
 - Program, 86
 - Script, 86
 - Template, 86
- Tang::SingletonObjectPool< T >, 88
 - get, 88
 - getInstance, 89
 - recycle, 89
- Tang::TangBase, 89
 - compileScript, 90
 - TangBase, 90
- Tang::TangScanner, 91
 - get_next_token, 92
 - TangScanner, 92
- TANG_UNUSED
 - macros.hpp, 112
- TangBase
 - Tang::TangBase, 90
- TangScanner
 - Tang::TangScanner, 92
- Template
 - Tang::Program, 86
 - test/test.cpp, 135
 - test/testSingletonObjectPool.cpp, 136