

Tang

0.1

Generated by Doxygen 1.9.1

1 Tang: A Template Language	1
1.1 Quick Description	1
1.2 Features	1
1.3 License	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 Tang::AstNode Class Reference	9
5.1.1 Detailed Description	11
5.1.2 Constructor & Destructor Documentation	11
5.1.2.1 AstNode()	11
5.1.3 Member Function Documentation	11
5.1.3.1 makeCopy()	11
5.2 Tang::AstNodeAdd Class Reference	12
5.2.1 Detailed Description	14
5.2.2 Constructor & Destructor Documentation	14
5.2.2.1 AstNodeAdd()	14
5.2.3 Member Function Documentation	14
5.2.3.1 makeCopy()	14
5.3 Tang::AstNodeDivide Class Reference	15
5.3.1 Detailed Description	17
5.3.2 Constructor & Destructor Documentation	17
5.3.2.1 AstNodeDivide()	17
5.3.3 Member Function Documentation	17
5.3.3.1 makeCopy()	17
5.4 Tang::AstNodeFloat Class Reference	18
5.4.1 Detailed Description	20
5.4.2 Constructor & Destructor Documentation	20
5.4.2.1 AstNodeFloat()	20
5.4.3 Member Function Documentation	20
5.4.3.1 makeCopy()	20
5.5 Tang::AstNodeInteger Class Reference	21
5.5.1 Detailed Description	23
5.5.2 Constructor & Destructor Documentation	23
5.5.2.1 AstNodeInteger()	23
5.5.3 Member Function Documentation	23

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

5.11.2.1 ComputedExpressionError()	42
5.11.3 Member Function Documentation	42
5.11.3.1 __add()	42
5.11.3.2 __divide()	43
5.11.3.3 __modulo()	43
5.11.3.4 __multiply()	44
5.11.3.5 __negative()	44
5.11.3.6 __subtract()	44
5.11.3.7 dump()	45
5.11.3.8 is_equal() [1/3]	45
5.11.3.9 is_equal() [2/3]	45
5.11.3.10 is_equal() [3/3]	46
5.11.3.11 makeCopy()	46
5.12 Tang::ComputedExpressionFloat Class Reference	47
5.12.1 Detailed Description	48
5.12.2 Constructor & Destructor Documentation	48
5.12.2.1 ComputedExpressionFloat()	48
5.12.3 Member Function Documentation	48
5.12.3.1 __add()	48
5.12.3.2 __divide()	49
5.12.3.3 __modulo()	49
5.12.3.4 __multiply()	50
5.12.3.5 __negative()	50
5.12.3.6 __subtract()	50
5.12.3.7 dump()	51
5.12.3.8 is_equal() [1/3]	51
5.12.3.9 is_equal() [2/3]	51
5.12.3.10 is_equal() [3/3]	52
5.12.3.11 makeCopy()	52
5.13 Tang::ComputedExpressionInteger Class Reference	53
5.13.1 Detailed Description	54
5.13.2 Constructor & Destructor Documentation	54
5.13.2.1 ComputedExpressionInteger()	54
5.13.3 Member Function Documentation	54
5.13.3.1 __add()	55
5.13.3.2 __divide()	55
5.13.3.3 __modulo()	55
5.13.3.4 __multiply()	56
5.13.3.5 __negative()	56
5.13.3.6 __subtract()	56
5.13.3.7 dump()	57
5.13.3.8 is_equal() [1/3]	57

5.13.3.9 is_equal() [2/3]	57
5.13.3.10 is_equal() [3/3]	58
5.13.3.11 makeCopy()	58
5.14 Tang::Error Class Reference	59
5.14.1 Detailed Description	60
5.14.2 Constructor & Destructor Documentation	60
5.14.2.1 Error() [1/2]	60
5.14.2.2 Error() [2/2]	60
5.14.3 Friends And Related Function Documentation	61
5.14.3.1 operator<<	61
5.15 Tang::GarbageCollected Class Reference	61
5.15.1 Detailed Description	63
5.15.2 Constructor & Destructor Documentation	63
5.15.2.1 GarbageCollected() [1/3]	63
5.15.2.2 GarbageCollected() [2/3]	63
5.15.2.3 ~GarbageCollected()	64
5.15.2.4 GarbageCollected() [3/3]	64
5.15.3 Member Function Documentation	64
5.15.3.1 make()	64
5.15.3.2 operator%()	65
5.15.3.3 operator*() [1/2]	65
5.15.3.4 operator*() [2/2]	66
5.15.3.5 operator+()	66
5.15.3.6 operator-() [1/2]	67
5.15.3.7 operator-() [2/2]	67
5.15.3.8 operator->()	68
5.15.3.9 operator/()	68
5.15.3.10 operator=() [1/2]	69
5.15.3.11 operator=() [2/2]	69
5.15.3.12 operator==(1/3)	70
5.15.3.13 operator==(2/3)	70
5.15.3.14 operator==(3/3)	71
5.15.4 Friends And Related Function Documentation	71
5.15.4.1 operator<<	71
5.16 Tang::location Class Reference	71
5.16.1 Detailed Description	73
5.17 Tang::position Class Reference	73
5.17.1 Detailed Description	74
5.18 Tang::Program Class Reference	74
5.18.1 Detailed Description	76
5.18.2 Member Enumeration Documentation	76
5.18.2.1 CodeType	76

5.18.3 Constructor & Destructor Documentation	76
5.18.3.1 Program()	76
5.18.4 Member Function Documentation	76
5.18.4.1 addBytecode()	77
5.18.4.2 dumpBytecode()	77
5.18.4.3 execute()	77
5.18.4.4 getAst()	77
5.18.4.5 getCode()	78
5.18.4.6 getResult()	78
5.19 Tang::SingletonObjectPool< T > Class Template Reference	78
5.19.1 Member Function Documentation	78
5.19.1.1 get()	79
5.19.1.2 getInstance()	79
5.19.1.3 recycle()	79
5.20 Tang::TangBase Class Reference	79
5.20.1 Detailed Description	80
5.20.2 Constructor & Destructor Documentation	80
5.20.2.1 TangBase()	80
5.20.3 Member Function Documentation	80
5.20.3.1 compileScript()	80
5.21 Tang::TangScanner Class Reference	81
5.21.1 Detailed Description	82
5.21.2 Constructor & Destructor Documentation	82
5.21.2.1 TangScanner()	82
5.21.3 Member Function Documentation	82
5.21.3.1 get_next_token()	82
6 File Documentation	83
6.1 build/generated/location.hh File Reference	83
6.1.1 Detailed Description	84
6.1.2 Function Documentation	84
6.1.2.1 operator<<() [1/2]	84
6.1.2.2 operator<<() [2/2]	85
6.2 include/astNode.hpp File Reference	85
6.2.1 Detailed Description	86
6.3 include/astNodeAdd.hpp File Reference	86
6.4 include/astNodeDivide.hpp File Reference	87
6.5 include/astNodeFloat.hpp File Reference	88
6.6 include/astNodeInteger.hpp File Reference	89
6.7 include/astNodeModulo.hpp File Reference	90
6.8 include/astNodeMultiply.hpp File Reference	91
6.9 include/astNodeNegative.hpp File Reference	92

6.10 include/astNodeSubtract.hpp File Reference	93
6.11 include/computedExpression.hpp File Reference	94
6.12 include/computedExpressionError.hpp File Reference	95
6.13 include/computedExpressionFloat.hpp File Reference	96
6.14 include/computedExpressionInteger.hpp File Reference	97
6.15 include/error.hpp File Reference	97
6.15.1 Detailed Description	98
6.16 include/garbageCollected.hpp File Reference	98
6.17 include/macros.hpp File Reference	99
6.17.1 Detailed Description	99
6.17.2 Macro Definition Documentation	100
6.17.2.1 TANG_UNUSED	100
6.18 include/opcode.hpp File Reference	100
6.18.1 Detailed Description	100
6.18.2 Enumeration Type Documentation	100
6.18.2.1 Opcode	100
6.19 include/program.hpp File Reference	101
6.19.1 Detailed Description	102
6.20 include/singletonObjectPool.hpp File Reference	102
6.21 include/tang.hpp File Reference	103
6.21.1 Detailed Description	104
6.22 include/tangBase.hpp File Reference	104
6.22.1 Detailed Description	105
6.23 include/tangScanner.hpp File Reference	105
6.23.1 Detailed Description	106
6.24 src/astNode.cpp File Reference	106
6.25 src/astNodeAdd.cpp File Reference	107
6.26 src/astNodeDivide.cpp File Reference	107
6.27 src/astNodeFloat.cpp File Reference	108
6.28 src/astNodeInteger.cpp File Reference	109
6.29 src/astNodeModulo.cpp File Reference	110
6.30 src/astNodeMultiply.cpp File Reference	111
6.31 src/astNodeNegative.cpp File Reference	112
6.32 src/astNodeSubtract.cpp File Reference	113
6.33 src/computedExpression.cpp File Reference	114
6.34 src/computedExpressionError.cpp File Reference	115
6.35 src/computedExpressionFloat.cpp File Reference	115
6.36 src/computedExpressionInteger.cpp File Reference	116
6.37 src/error.cpp File Reference	117
6.37.1 Function Documentation	117
6.37.1.1 operator<<()	117
6.38 src/program-dumpBytecode.cpp File Reference	118

6.38.1 Macro Definition Documentation	118
6.38.1.1 DUMPPROGRAMCHECK	118
6.39 src/program-execute.cpp File Reference	119
6.39.1 Macro Definition Documentation	119
6.39.1.1 EXECUTEPROGRAMCHECK	119
6.39.1.2 STACKCHECK	120
6.40 src/program.cpp File Reference	120
6.41 src/tangBase.cpp File Reference	121
6.42 test/test.cpp File Reference	121
6.42.1 Detailed Description	122
6.43 test/testSingletonObjectPool.cpp File Reference	122
Index	123

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	12
Tang::AstNodeDivide	15
Tang::AstNodeFloat	18
Tang::AstNodeInteger	21
Tang::AstNodeModulo	24
Tang::AstNodeMultiply	27
Tang::AstNodeNegative	30
Tang::AstNodeSubtract	33
Tang::ComputedExpression	36
Tang::ComputedExpressionError	41
Tang::ComputedExpressionFloat	47
Tang::ComputedExpressionInteger	53
Tang::Error	59
Tang::GarbageCollected	61
Tang::location	71
Tang::position	73
Tang::Program	74
Tang::SingletonObjectPool< T >	78
Tang::TangBase	79
TangTangFlexLexer	
Tang::TangScanner	81

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	12
Tang::AstNodeDivide	An AstNode that represents a "/" expression	15
Tang::AstNodeFloat	An AstNode that represents an float literal	18
Tang::AstNodeInteger	An AstNode that represents an integer literal	21
Tang::AstNodeModulo	An AstNode that represents a "%" expression	24
Tang::AstNodeMultiply	An AstNode that represents a "*" expression	27
Tang::AstNodeNegative	An AstNode that represents a unary negation	30
Tang::AstNodeSubtract	An AstNode that represents a "-" expression	33
Tang::ComputedExpression	Represents the result of a computation that has been executed	36
Tang::ComputedExpressionError	Represents a Runtime Error	41
Tang::ComputedExpressionFloat	Represents a Float that is the result of a computation	47
Tang::ComputedExpressionInteger	Represents an Integer that is the result of a computation	53
Tang::Error	Used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error	59
Tang::GarbageCollected	A container that acts as a resource-counting garbage collector for the specified type	61
Tang::location	Two points in a source file	71
Tang::position	A point in a source file	73

Tang::Program	
Represents a compiled script or template that may be executed	74
Tang::SingletonObjectPool< T >	78
Tang::TangBase	
The base class for the Tang programming language	79
Tang::TangScanner	
The Flex lexer class for the main Tang language	81

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 <code>Tang::location</code> class	83
include/astNode.hpp	
Define the <code>Tang::AstNode</code> and its associated/derivative classes	85
include/astNodeAdd.hpp	86
include/astNodeDivide.hpp	87
include/astNodeFloat.hpp	88
include/astNodeInteger.hpp	89
include/astNodeModulo.hpp	90
include/astNodeMultiply.hpp	91
include/astNodeNegative.hpp	92
include/astNodeSubtract.hpp	93
include/computedExpression.hpp	94
include/computedExpressionError.hpp	95
include/computedExpressionFloat.hpp	96
include/computedExpressionInteger.hpp	97
include/error.hpp	
Define the <code>Tang::Error</code> class used to describe syntax and runtime errors	97
include/garbageCollected.hpp	98
include/macros.hpp	
Contains generic macros	99
include/opcode.hpp	
Declare the Opcodes used in the Bytecode representation of a program	100
include/program.hpp	
Define the <code>Tang::Program</code> class used to compile and execute source code	101
include/singletonObjectPool.hpp	102
include/tang.hpp	
Header file supplied for use by 3rd party code so that they can easily include all necessary headers	103
include/tangBase.hpp	
Defines the <code>Tang::TangBase</code> class used to interact with Tang	104
include/tangScanner.hpp	
Defines the <code>Tang::TangScanner</code> used to tokenize a Tang script	105
src/astNode.cpp	106
src/astNodeAdd.cpp	107

src/ astNodeDivide.cpp	107
src/ astNodeFloat.cpp	108
src/ astNodeInteger.cpp	109
src/ astNodeModulo.cpp	110
src/ astNodeMultiply.cpp	111
src/ astNodeNegative.cpp	112
src/ astNodeSubtract.cpp	113
src/ computedExpression.cpp	114
src/ computedExpressionError.cpp	115
src/ computedExpressionFloat.cpp	115
src/ computedExpressionInteger.cpp	116
src/ error.cpp	117
src/ program-dumpBytecode.cpp	118
src/ program-execute.cpp	119
src/ program.cpp	120
src/ tangBase.cpp	121
test/ test.cpp	
Test the general language behaviors	121
test/ testSingletonObjectPool.cpp	122

Chapter 5

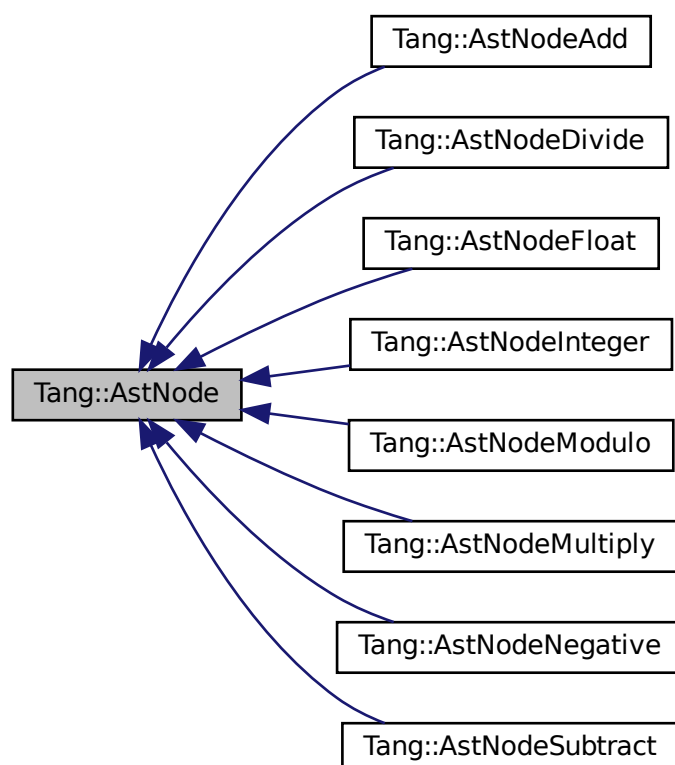
Class Documentation

5.1 Tang::AstNode Class Reference

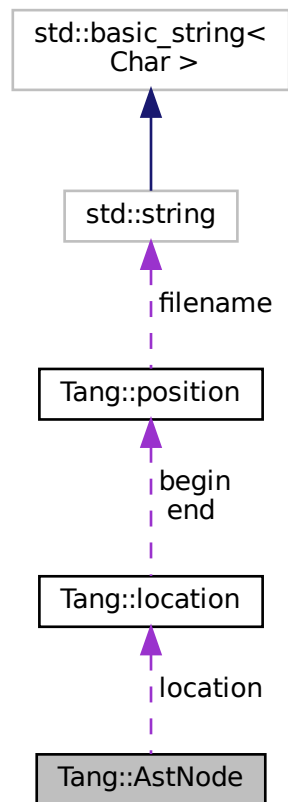
Base class for representing nodes of an Abstract Syntax Tree (AST).

```
#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

<code>loc</code>	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](#), and [Tang::AstNodeAdd](#).

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

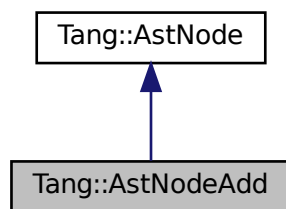
- `include/astNode.hpp`
- `src/astNode.cpp`

5.2 Tang::AstNodeAdd Class Reference

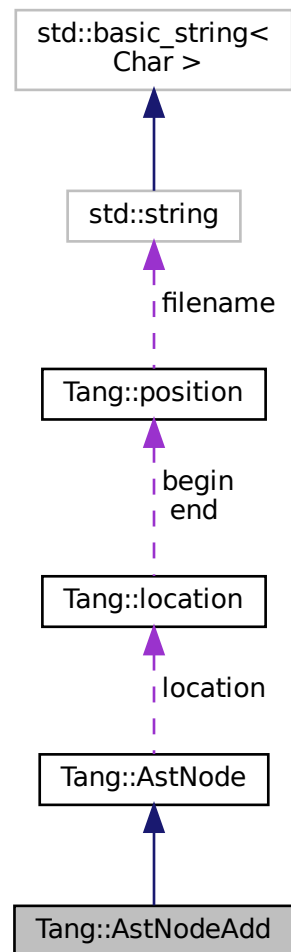
An [AstNode](#) that represents a "+" expression.

```
#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:

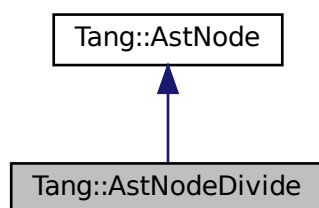
- include/[astNodeAdd.hpp](#)
- src/[astNodeAdd.cpp](#)

5.3 Tang::AstNodeDivide Class Reference

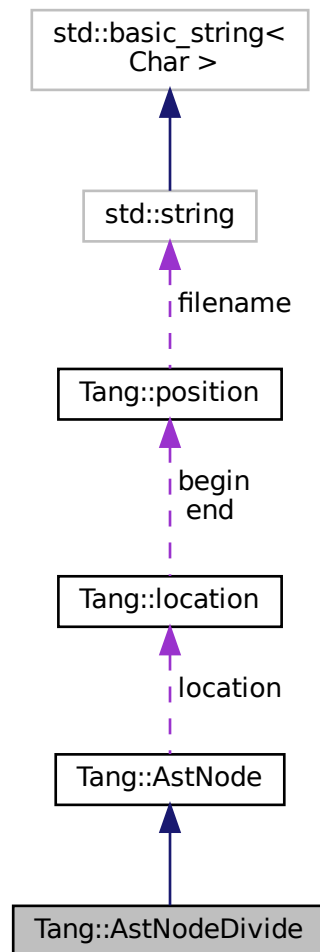
An [AstNode](#) that represents a "/" expression.

```
#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.

5.3.1 Detailed Description

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

5.3.2 Constructor & Destructor Documentation

5.3.2.1 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.

5.3.3 Member Function Documentation

5.3.3.1 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:

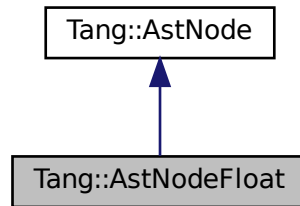
- include/[astNodeDivide.hpp](#)
- src/[astNodeDivide.cpp](#)

5.4 Tang::AstNodeFloat Class Reference

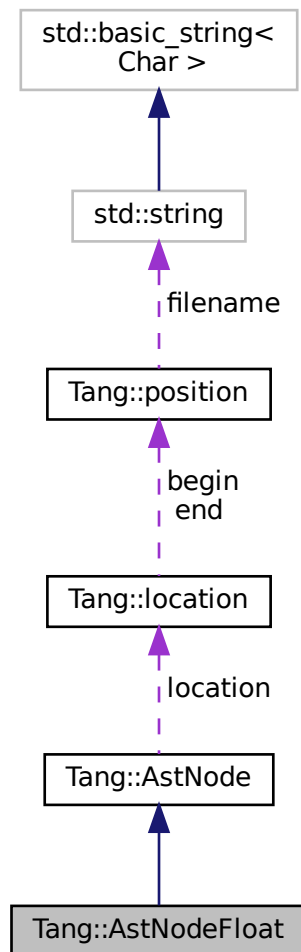
An [AstNode](#) that represents an float literal.

```
#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.

5.4.1 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.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 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.

5.4.3 Member Function Documentation

5.4.3.1 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:

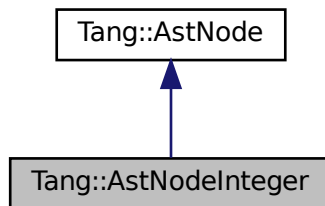
- include/[astNodeFloat.hpp](#)
- src/[astNodeFloat.cpp](#)

5.5 Tang::AstNodeInteger Class Reference

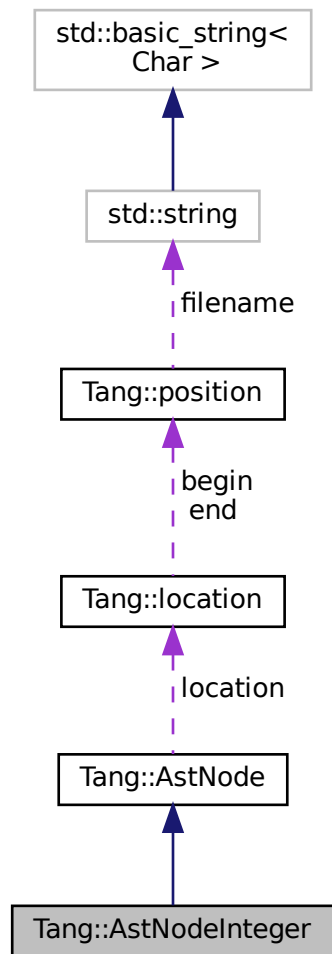
An [AstNode](#) that represents an integer literal.

```
#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.

5.5.1 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.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 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.

5.5.3 Member Function Documentation

5.5.3.1 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:

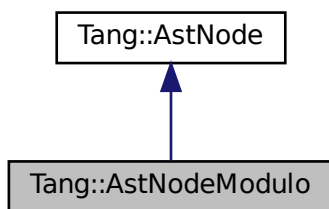
- [include/astNodeInteger.hpp](#)
- [src/astNodeInteger.cpp](#)

5.6 Tang::AstNodeModulo Class Reference

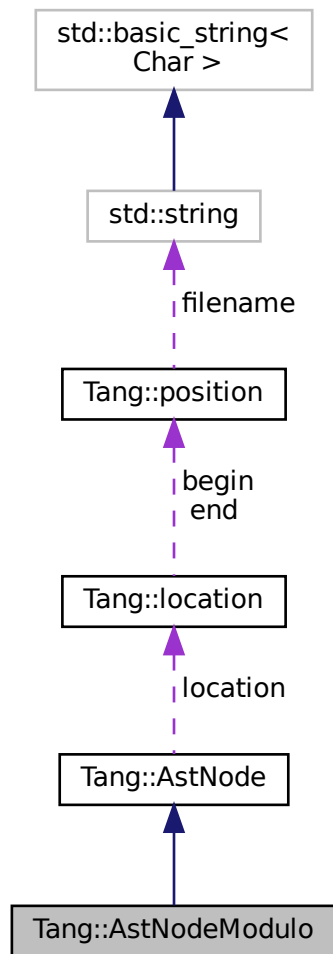
An [AstNode](#) that represents a "%" expression.

```
#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.

5.6.1 Detailed Description

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

5.6.2 Constructor & Destructor Documentation

5.6.2.1 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.

5.6.3 Member Function Documentation

5.6.3.1 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:

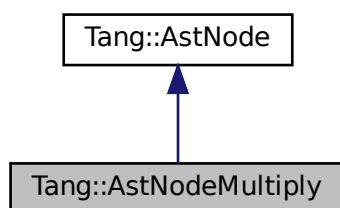
- include/[astNodeModulo.hpp](#)
- src/[astNodeModulo.cpp](#)

5.7 Tang::AstNodeMultiply Class Reference

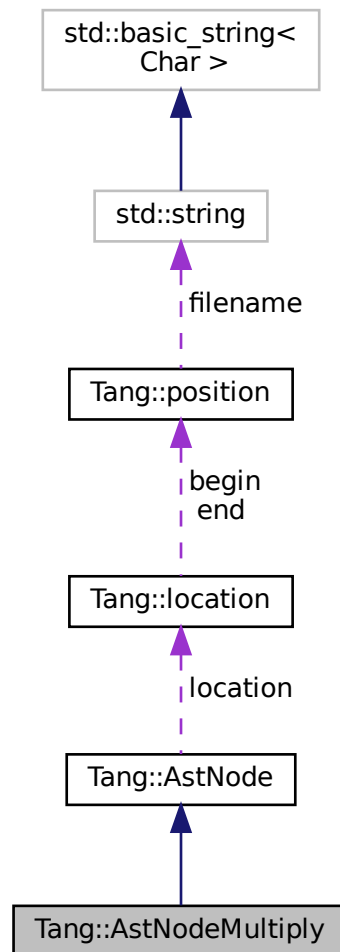
An [AstNode](#) that represents a "*" expression.

```
#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.

5.7.1 Detailed Description

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

5.7.2 Constructor & Destructor Documentation

5.7.2.1 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.

5.7.3 Member Function Documentation

5.7.3.1 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:

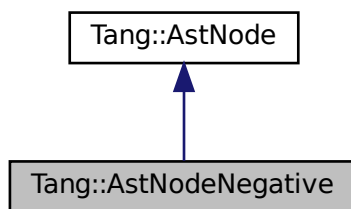
- include/[astNodeMultiply.hpp](#)
- src/[astNodeMultiply.cpp](#)

5.8 Tang::AstNodeNegative Class Reference

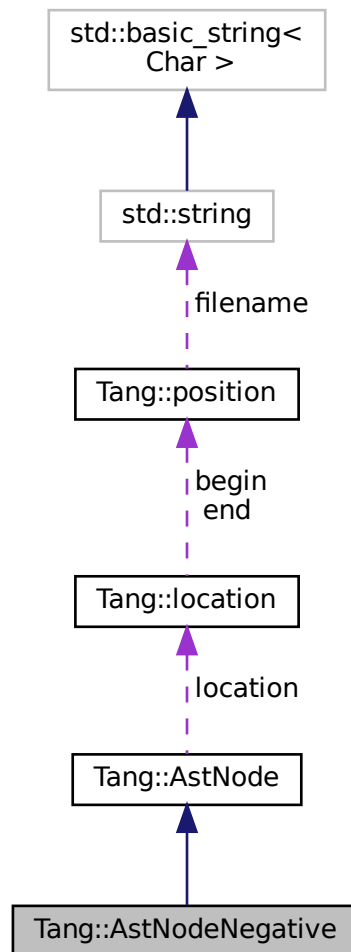
An [AstNode](#) that represents a unary negation.

```
#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.

5.8.1 Detailed Description

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

5.8.2 Constructor & Destructor Documentation

5.8.2.1 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.

5.8.3 Member Function Documentation

5.8.3.1 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:

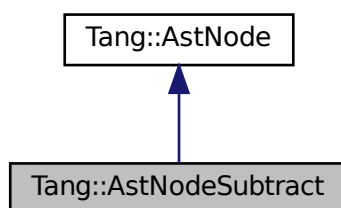
- include/[astNodeNegative.hpp](#)
- src/[astNodeNegative.cpp](#)

5.9 Tang::AstNodeSubtract Class Reference

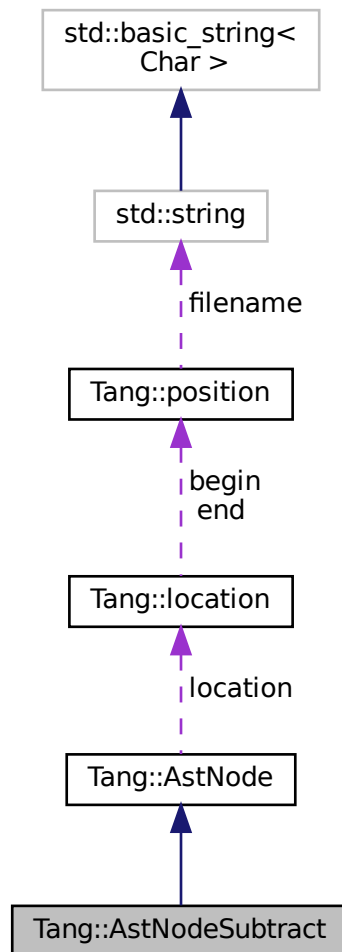
An [AstNode](#) that represents a "-" expression.

```
#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.

5.9.1 Detailed Description

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

5.9.2 Constructor & Destructor Documentation

5.9.2.1 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.

5.9.3 Member Function Documentation

5.9.3.1 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:

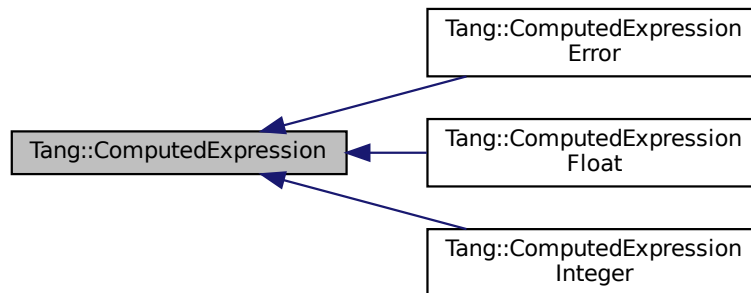
- include/[astNodeSubtract.hpp](#)
- src/[astNodeSubtract.cpp](#)

5.10 Tang::ComputedExpression Class Reference

Represents the result of a computation that has been executed.

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

Inheritance diagram for Tang::ComputedExpression:



Public Member Functions

- virtual [~ComputedExpression](#) ()
The object destructor.
- virtual std::string [dump](#) () const
Output the contents of the [ComputedExpression](#) as a string.
- virtual [ComputedExpression](#) * [makeCopy](#) () const
Make a copy of the [ComputedExpression](#) (recursively, if appropriate).
- virtual bool [is_equal](#) (const int &val) const
Check whether or not the computed expression is equal to another value.
- virtual bool [is_equal](#) (const double &val) const
Check whether or not the computed expression is equal to another value.
- virtual bool [is_equal](#) (const [Error](#) &val) const
Check whether or not the computed expression is equal to another value.
- virtual [GarbageCollected](#) [__add](#) (const [GarbageCollected](#) &rhs) const
Compute the result of adding this value and the supplied value.
- virtual [GarbageCollected](#) [__subtract](#) (const [GarbageCollected](#) &rhs) const
Compute the result of subtracting this value and the supplied value.
- virtual [GarbageCollected](#) [__multiply](#) (const [GarbageCollected](#) &rhs) const
Compute the result of multiplying this value and the supplied value.
- virtual [GarbageCollected](#) [__divide](#) (const [GarbageCollected](#) &rhs) const
Compute the result of dividing this value and the supplied value.
- virtual [GarbageCollected](#) [__modulo](#) (const [GarbageCollected](#) &rhs) const
Compute the result of moduloing this value and the supplied value.
- virtual [GarbageCollected](#) [__negative](#) () const
Compute the result of negating this value.

5.10.1 Detailed Description

Represents the result of a computation that has been executed.

5.10.2 Member Function Documentation

5.10.2.1 __add()

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

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

Parameters

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

Returns

The result of the operation.

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

5.10.2.2 __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](#).

5.10.2.3 `__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](#).

5.10.2.4 `__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](#).

5.10.2.5 `__negative()`

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

Compute the result of negating this value.

Returns

The result of the operation.

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

5.10.2.6 __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](#).

5.10.2.7 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](#).

5.10.2.8 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](#).

5.10.2.9 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](#).

5.10.2.10 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](#).

5.10.2.11 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:

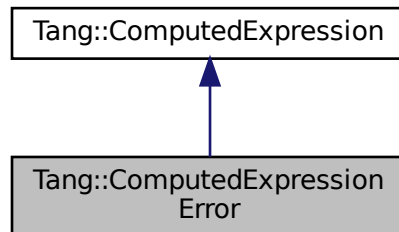
- include/[computedExpression.hpp](#)
- src/[computedExpression.cpp](#)

5.11 Tang::ComputedExpressionError Class Reference

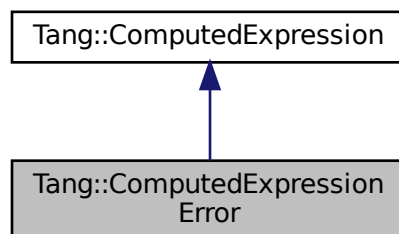
Represents a Runtime [Error](#).

```
#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.

5.11.1 Detailed Description

Represents a Runtime `Error`.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 ComputedExpressionError()

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

Construct a Runtime `Error`.

Parameters

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

5.11.3 Member Function Documentation

5.11.3.1 __add()

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

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

Parameters

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

Returns

The result of the operation.

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

5.11.3.2 __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](#).

5.11.3.3 __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](#).

5.11.3.4 `__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](#).

5.11.3.5 `__negative()`

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

Compute the result of negating this value.

Returns

The result of the operation.

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

5.11.3.6 `__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](#).

5.11.3.7 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](#).

5.11.3.8 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](#).

5.11.3.9 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](#).

5.11.3.10 `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](#).

5.11.3.11 `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:

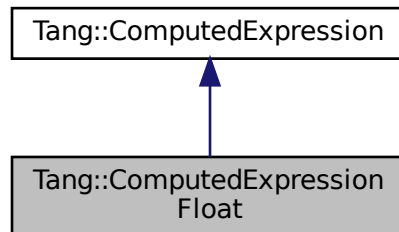
- [include/computedExpressionError.hpp](#)
- [src/computedExpressionError.cpp](#)

5.12 Tang::ComputedExpressionFloat Class Reference

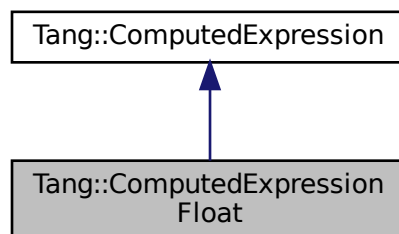
Represents a Float that is the result of a computation.

```
#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 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`

5.12.1 Detailed Description

Represents a Float that is the result of a computation.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 ComputedExpressionFloat()

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

Construct a Float result.

Parameters

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

5.12.3 Member Function Documentation

5.12.3.1 __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](#).

5.12.3.2 __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](#).

5.12.3.3 __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](#).

5.12.3.4 `__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](#).

5.12.3.5 `__negative()`

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

Compute the result of negating this value.

Returns

The result of the operation.

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

5.12.3.6 `__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](#).

5.12.3.7 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](#).

5.12.3.8 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](#).

5.12.3.9 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](#).

5.12.3.10 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](#).

5.12.3.11 makeCopy()

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

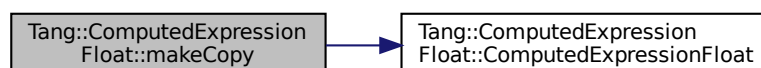
Make a copy of the [ComputedExpression](#) (recursively, if appropriate).

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:

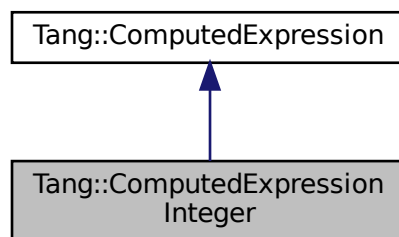
- [include/computedExpressionFloat.hpp](#)
- [src/computedExpressionFloat.cpp](#)

5.13 Tang::ComputedExpressionInteger Class Reference

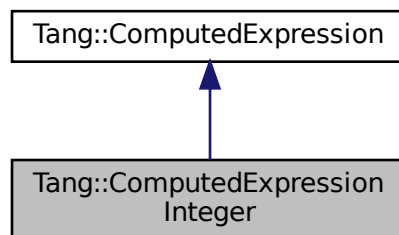
Represents an Integer that is the result of a computation.

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

Friends

- class `ComputedExpressionFloat`

5.13.1 Detailed Description

Represents an Integer that is the result of a computation.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 `ComputedExpressionInteger()`

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

Construct an Integer result.

Parameters

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

5.13.3 Member Function Documentation

5.13.3.1 __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](#).

5.13.3.2 __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](#).

5.13.3.3 __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](#).

5.13.3.4 __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](#).

5.13.3.5 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

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

5.13.3.6 __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](#).

5.13.3.7 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](#).

5.13.3.8 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](#).

5.13.3.9 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](#).

5.13.3.10 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](#).

5.13.3.11 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:

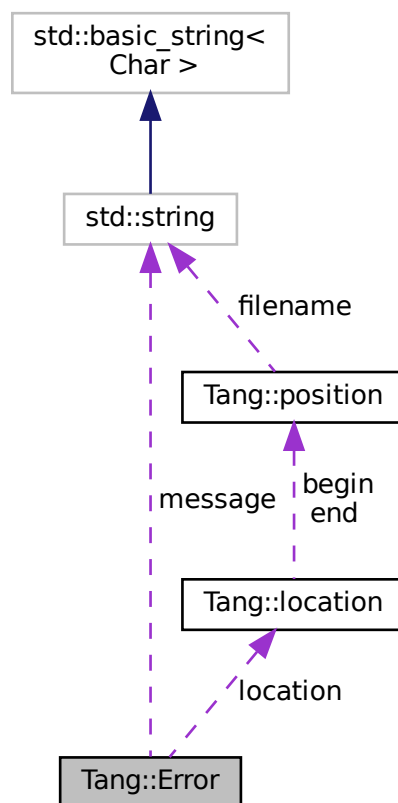
- [include/computedExpressionInteger.hpp](#)
- [src/computedExpressionInteger.cpp](#)

5.14 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.

5.14.1 Detailed Description

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

5.14.2 Constructor & Destructor Documentation

5.14.2.1 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.
----------------	--------------------------------

5.14.2.2 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.

5.14.3 Friends And Related Function Documentation

5.14.3.1 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:

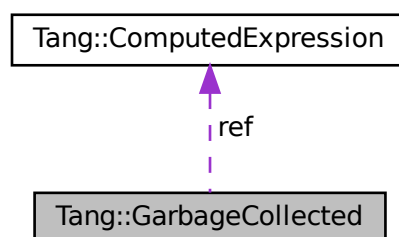
- [include/error.hpp](#)
- [src/error.cpp](#)

5.15 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.

5.15.1 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.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 GarbageCollected() [1/3]

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

Copy Constructor.

Parameters

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

5.15.2.2 GarbageCollected() [2/3]

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

Move Constructor.

Parameters

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

5.15.2.3 ~GarbageCollected()

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

Destructor.

Clean up the tracked object, if appropriate.

5.15.2.4 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.
-----------------	---

5.15.3 Member Function Documentation**5.15.3.1 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:



5.15.3.2 operator%()

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

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

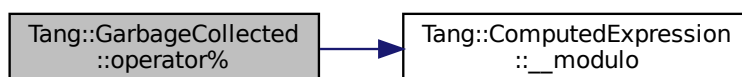
Parameters

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

Returns

The result of the operation.

Here is the call graph for this function:



5.15.3.3 operator*() [1/2]

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

Access the tracked object.

Returns

A reference to the tracked object.

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

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

Perform a multiplication between two `GarbageCollected` values.

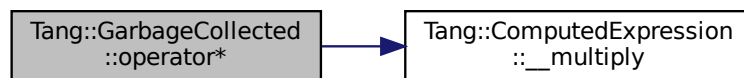
Parameters

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

Returns

The result of the operation.

Here is the call graph for this function:



5.15.3.5 `operator+()`

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

Perform an addition between two `GarbageCollected` values.

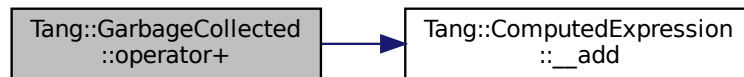
Parameters

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

Returns

The result of the operation.

Here is the call graph for this function:

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

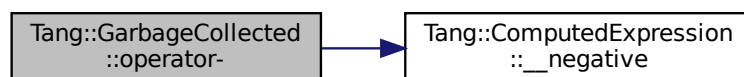
```
GarbageCollected GarbageCollected::operator- ( ) const
```

Perform a negation on the `GarbageCollected` value.

Returns

The result of the operation.

Here is the call graph for this function:

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

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

Perform a subtraction between two `GarbageCollected` values.

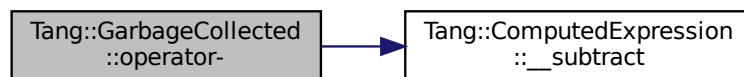
Parameters

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

Returns

The result of the operation.

Here is the call graph for this function:



5.15.3.8 operator->()

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

Access the tracked object as a pointer.

Returns

A pointer to the tracked object.

5.15.3.9 operator/()

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

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

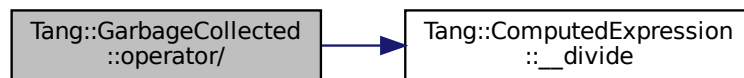
Parameters

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

Returns

The result of the operation.

Here is the call graph for this function:



5.15.3.10 operator=() [1/2]

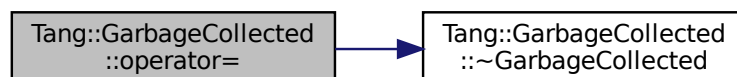
```
GarbageCollected& Tang::GarbageCollected::operator= (
    const GarbageCollected & other ) [inline]
```

Copy Assignment.

Parameters

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

Here is the call graph for this function:



5.15.3.11 operator=() [2/2]

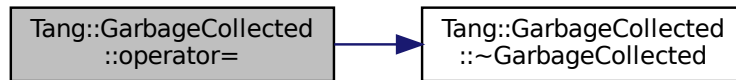
```
GarbageCollected& Tang::GarbageCollected::operator= (
    GarbageCollected && other ) [inline]
```

Move Assignment.

Parameters

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

Here is the call graph for this function:



5.15.3.12 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.

5.15.3.13 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.

5.15.3.14 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.

5.15.4 Friends And Related Function Documentation**5.15.4.1 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:

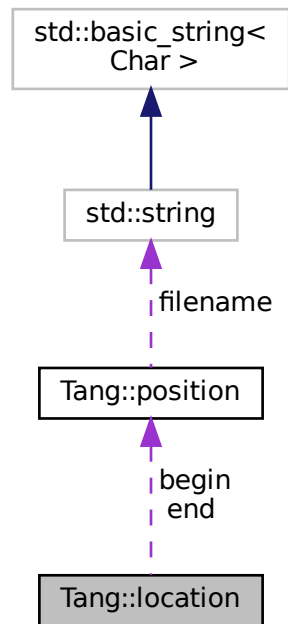
- include/[garbageCollected.hpp](#)
- src/garbageCollected.cpp

5.16 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.

5.16.1 Detailed Description

Two points in a source file.

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

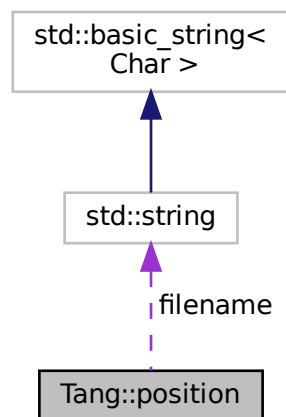
- build/generated/[location.hh](#)

5.17 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.

5.17.1 Detailed Description

A point in a source file.

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

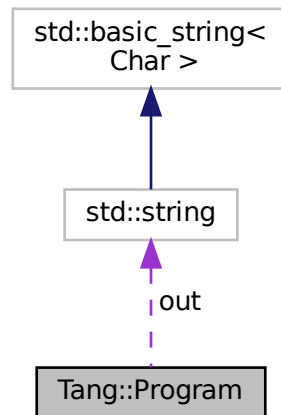
- `build/generated/location.hh`

5.18 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.

5.18.1 Detailed Description

Represents a compiled script or template that may be executed.

5.18.2 Member Enumeration Documentation

5.18.2.1 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.

5.18.3 Constructor & Destructor Documentation

5.18.3.1 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> .

5.18.4 Member Function Documentation

5.18.4.1 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.
-----------	-----------------------------------

5.18.4.2 dumpBytecode()

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

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

Returns

A string containing the Opcode representation.

5.18.4.3 execute()

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

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

Returns

The current [Program](#) object.

5.18.4.4 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.

5.18.4.5 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.

5.18.4.6 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](#)

5.19 [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.

5.19.1 Member Function Documentation

5.19.1.1 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.

5.19.1.2 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.

5.19.1.3 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](#)

5.20 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](#).

5.20.1 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).

5.20.2 Constructor & Destructor Documentation

5.20.2.1 TangBase()

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

The constructor.

Isn't it glorious.

5.20.3 Member Function Documentation

5.20.3.1 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:

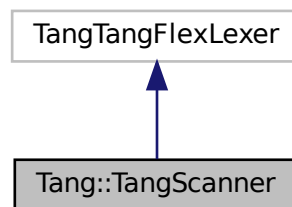
- include/[tangBase.hpp](#)
- src/[tangBase.cpp](#)

5.21 Tang::TangScanner Class Reference

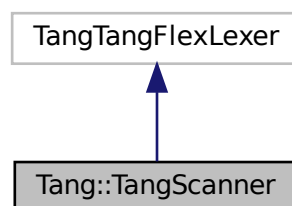
The Flex lexer class for the main Tang language.

```
#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.

5.21.1 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.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 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)

5.21.3 Member Function Documentation

5.21.3.1 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

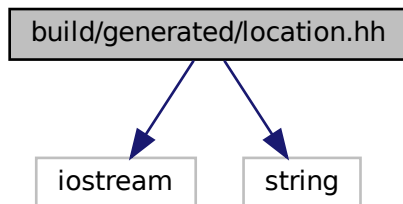
6.1 build/generated/location.hh File Reference

Define the Tang ::location class.

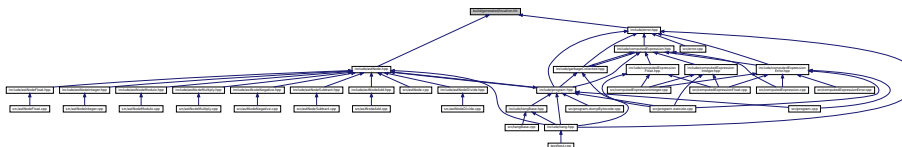
```
#include <iostream>
```

```
#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.

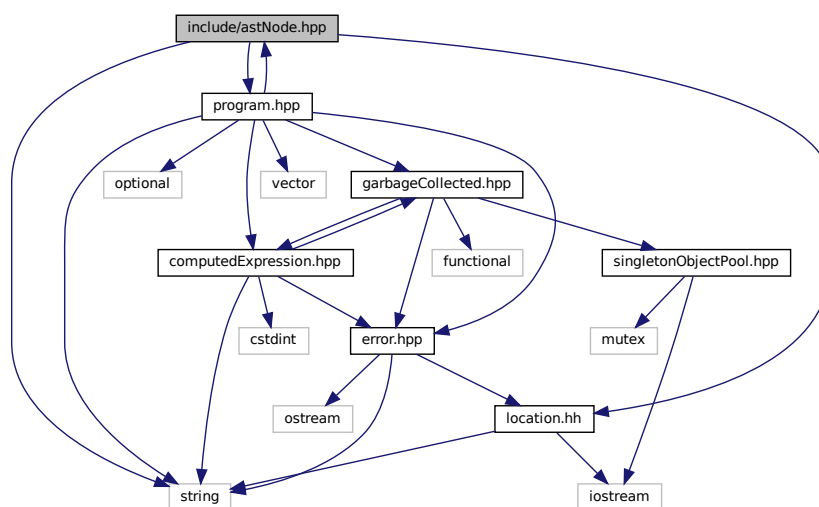
Parameters

<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:
```

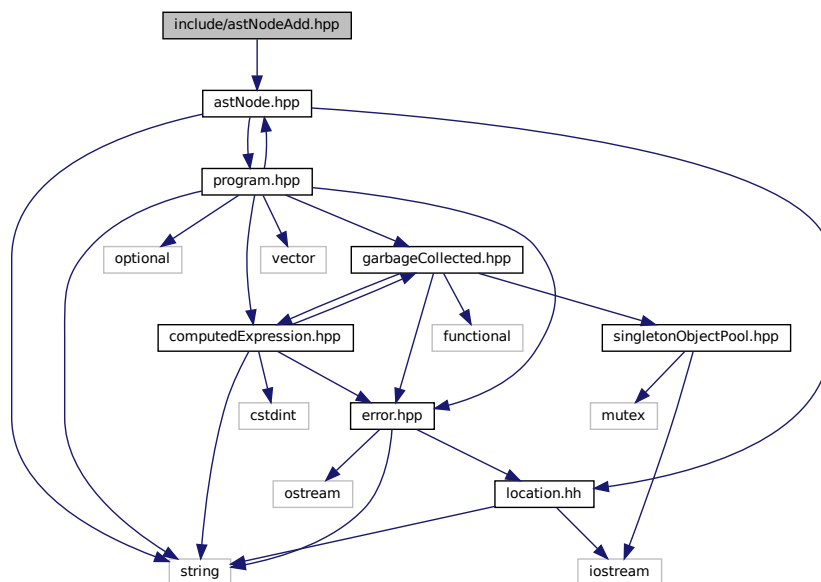


The diagram illustrates a hierarchical tree structure for classifying 1000 samples. The root node is labeled '1000samples'. It branches into 10 intermediate nodes, which then branch into 100 leaf nodes. The leaf nodes are labeled with sample IDs, such as '1000samples_1', '1000samples_2', ..., '1000samples_1000'.

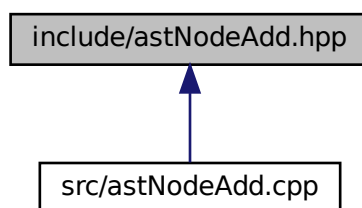
- class `Tang::AstNode`
Base class for representing nodes of an Abstract Syntax Tree (AST).

Define the `Tang::AstNode` and its associated/derivative classes.

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



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



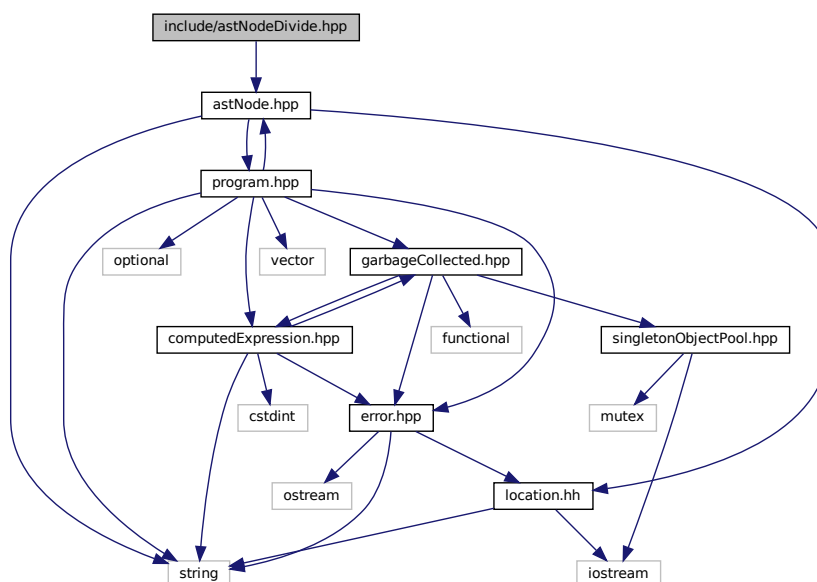
Classes

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

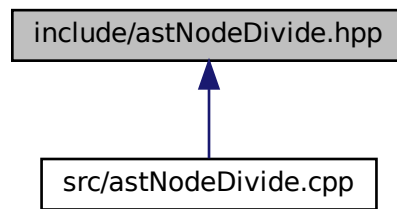
6.4 include/astNodeDivide.hpp File Reference

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

Include dependency graph for astNodeDivide.hpp:



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



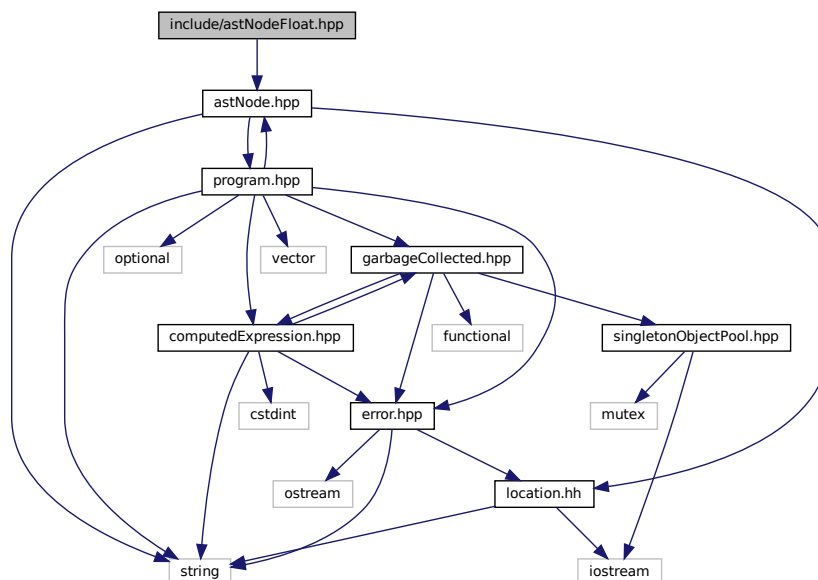
Classes

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

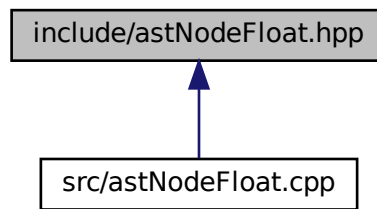
6.5 include/astNodeFloat.hpp File Reference

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

Include dependency graph for astNodeFloat.hpp:



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



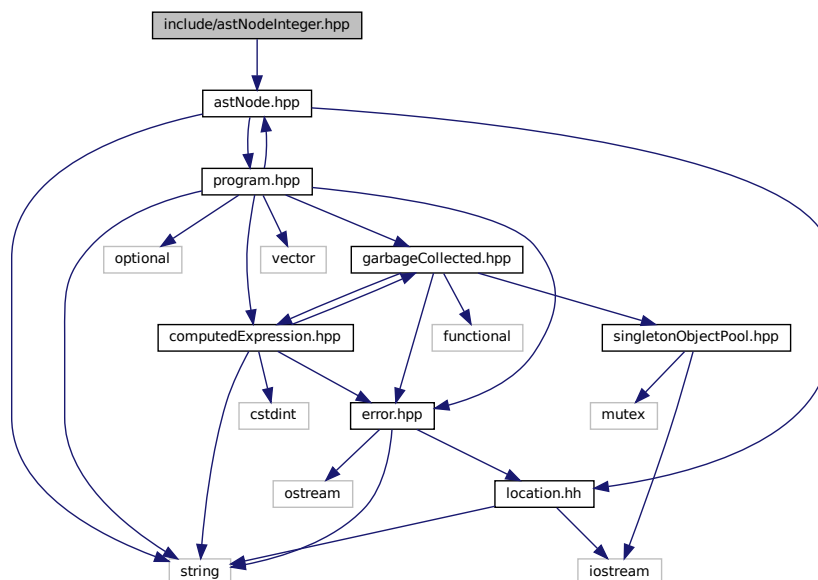
Classes

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

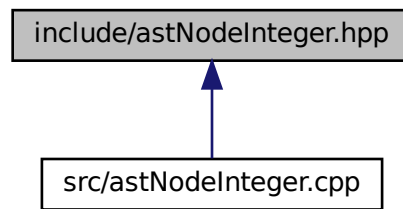
6.6 include/astNodeInteger.hpp File Reference

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

Include dependency graph for `astNodeInteger.hpp`:



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



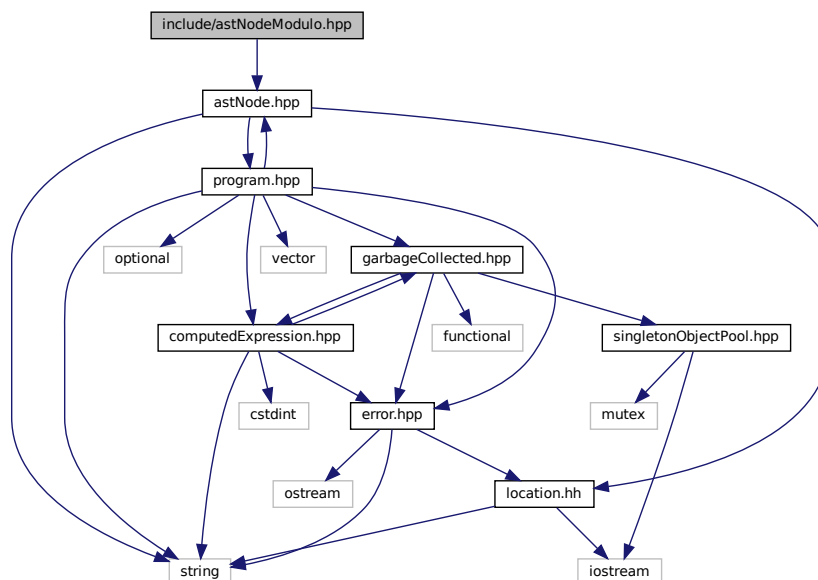
Classes

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

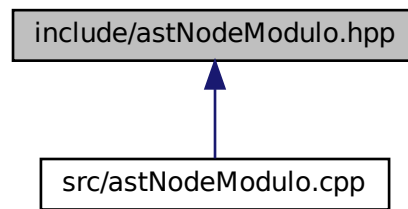
6.7 include/astNodeModulo.hpp File Reference

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

Include dependency graph for astNodeModulo.hpp:



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



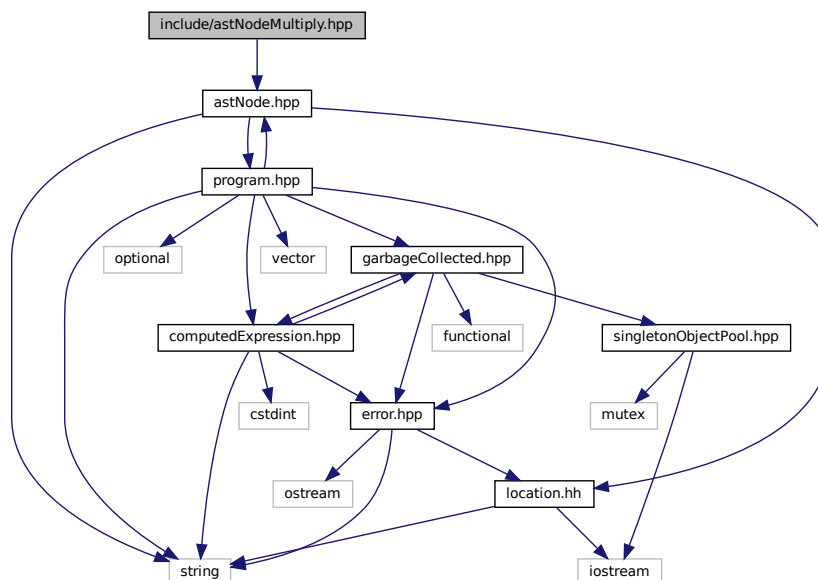
Classes

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

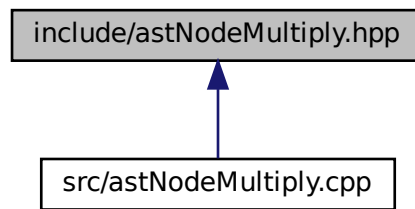
6.8 include/astNodeMultiply.hpp File Reference

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

Include dependency graph for astNodeMultiply.hpp:



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



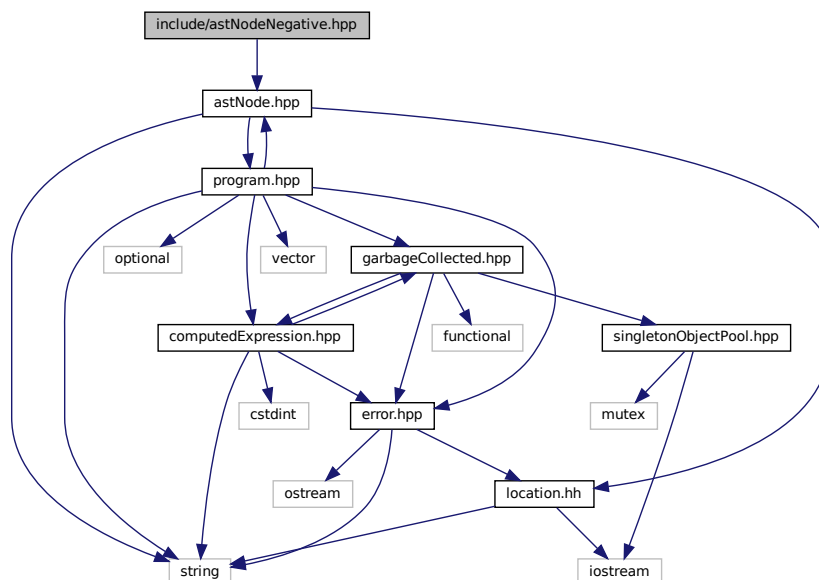
Classes

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

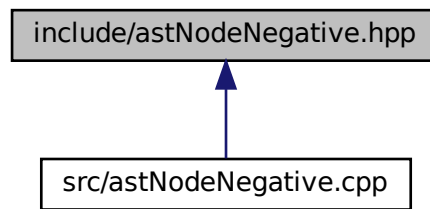
6.9 include/astNodeNegative.hpp File Reference

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

Include dependency graph for astNodeNegative.hpp:



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



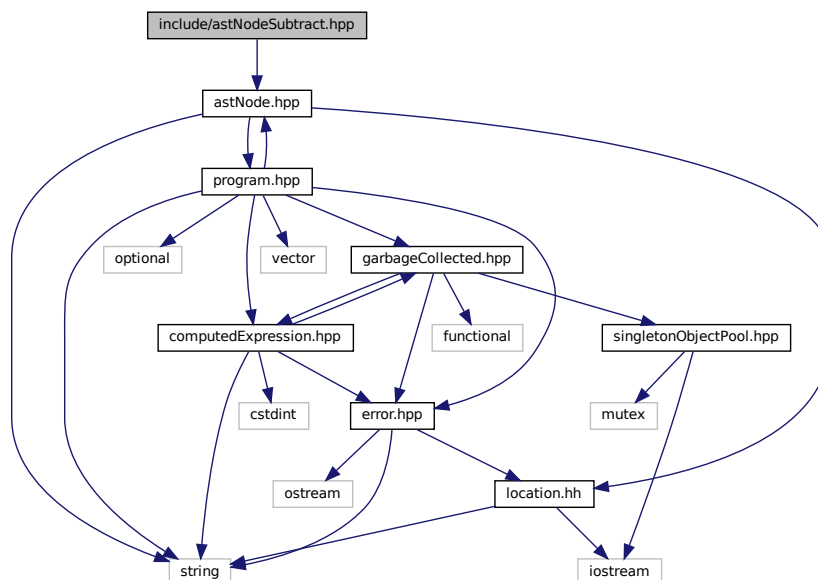
Classes

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

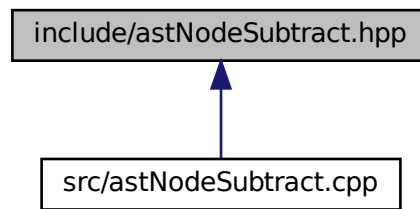
6.10 include/astNodeSubtract.hpp File Reference

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

Include dependency graph for `astNodeSubtract.hpp`:



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

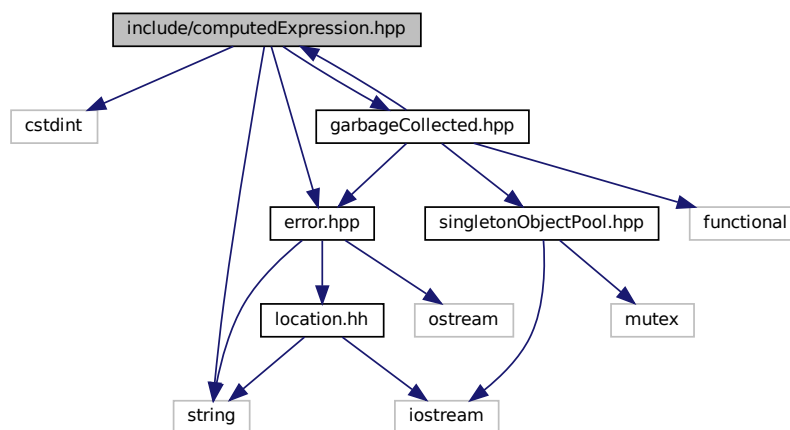


Classes

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

6.11 include/computedExpression.hpp File Reference

```
#include <cstdint>
#include <string>
#include "garbageCollected.hpp"
#include "error.hpp"
Include dependency graph for computedExpression.hpp:
```



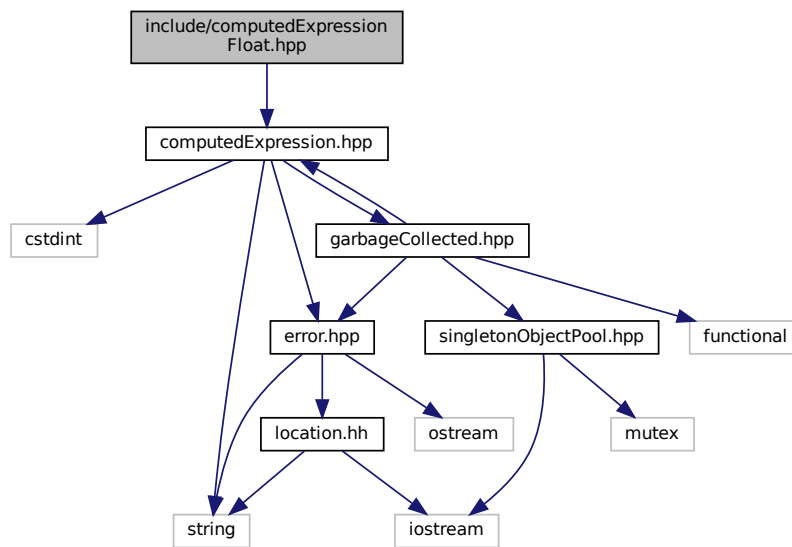
Classes

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

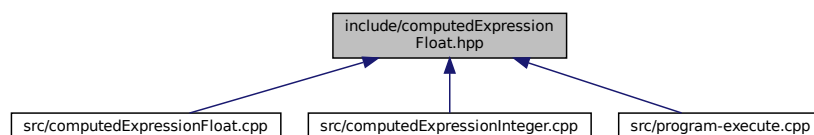
6.13 include/computedExpressionFloat.hpp File Reference

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

Include dependency graph for computedExpressionFloat.hpp:



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



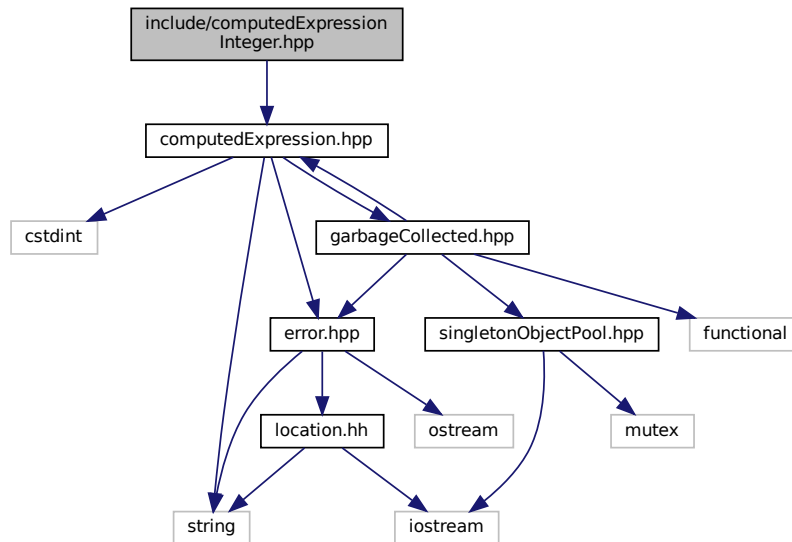
Classes

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

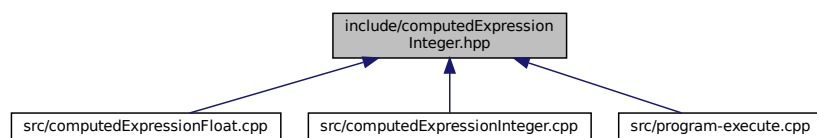
6.14 include/computedExpressionInteger.hpp File Reference

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

Include dependency graph for computedExpressionInteger.hpp:



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



Classes

- class [Tang::ComputedExpressionInteger](#)
Represents an Integer that is the result of a computation.

6.15 include/error.hpp File Reference

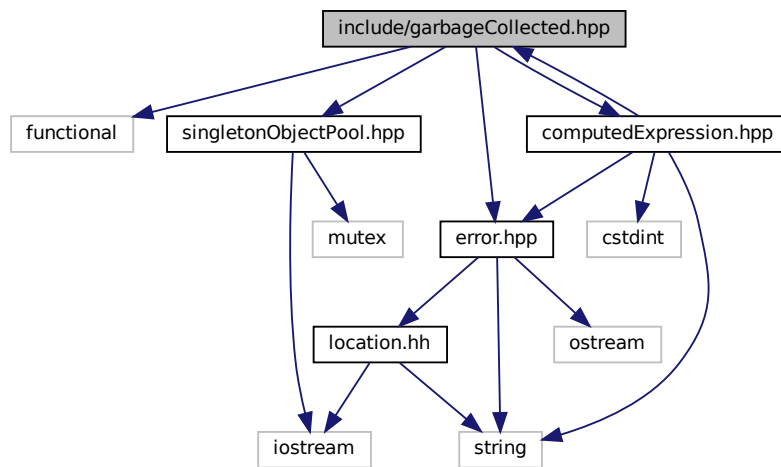
Define the [Tang::Error](#) class used to describe syntax and runtime errors.

```
#include <string>
#include <ostream>
```

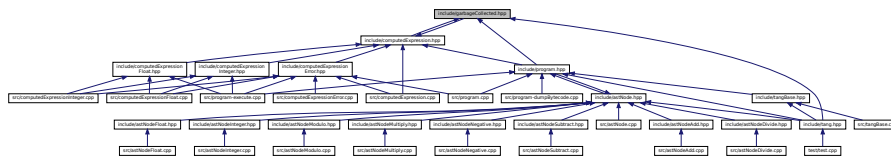


```
#include "error.hpp"
```

Include dependency graph for garbageCollected.hpp:



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



Classes

- class `Tang::GarbageCollected`

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

6.17 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.

6.17.1 Detailed Description

Contains generic macros.

6.17.2 Macro Definition Documentation

6.17.2.1 TANG_UNUSED

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

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

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

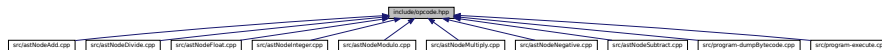
Parameters

x	The argument to be ignored.
---	-----------------------------

6.18 include/opcode.hpp File Reference

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

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



Enumerations

- enum class [Tang::Opcode](#) {
[INTEGER](#) , [FLOAT](#) , [ADD](#) , [SUBTRACT](#) ,
[MULTIPLY](#) , [DIVIDE](#) , [MODULO](#) , [NEGATIVE](#) }

6.18.1 Detailed Description

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

6.18.2 Enumeration Type Documentation

6.18.2.1 Opcode

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

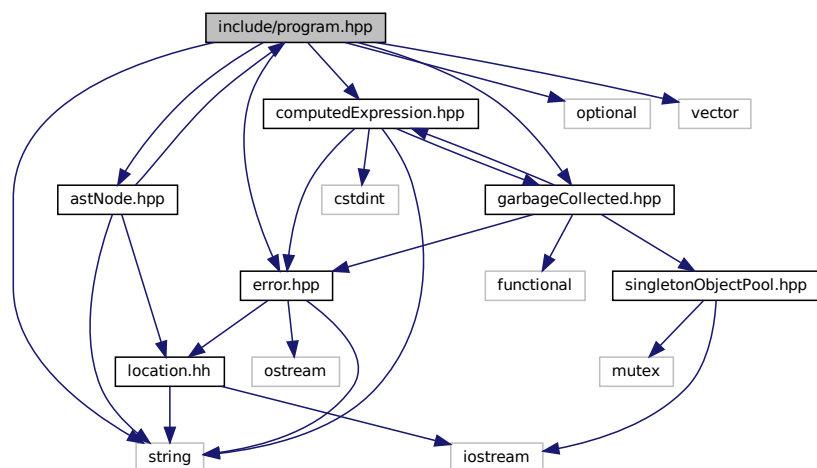
Enumerator

INTEGER	Push an integer onto the stack.
FLOAT	Push a floating point number onto the stack.
ADD	Pop rhs, pop lhs, push lhs + rhs.
SUBTRACT	Pop rhs, pop lhs, push lhs - rhs.
MULTIPLY	Pop rhs, pop lhs, push lhs * rhs.
DIVIDE	Pop rhs, pop lhs, push lhs / rhs.
MODULO	Pop rhs, pop lhs, push lhs % rhs.
NEGATIVE	Pop val, push negative val.

6.19 include/program.hpp File Reference

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

```
#include <string>
#include <optional>
#include <vector>
#include "astNode.hpp"
#include "error.hpp"
#include "computedExpression.hpp"
#include "garbageCollected.hpp"
Include dependency graph for program.hpp:
```



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



Macros

- #define [GROW](#) 1024

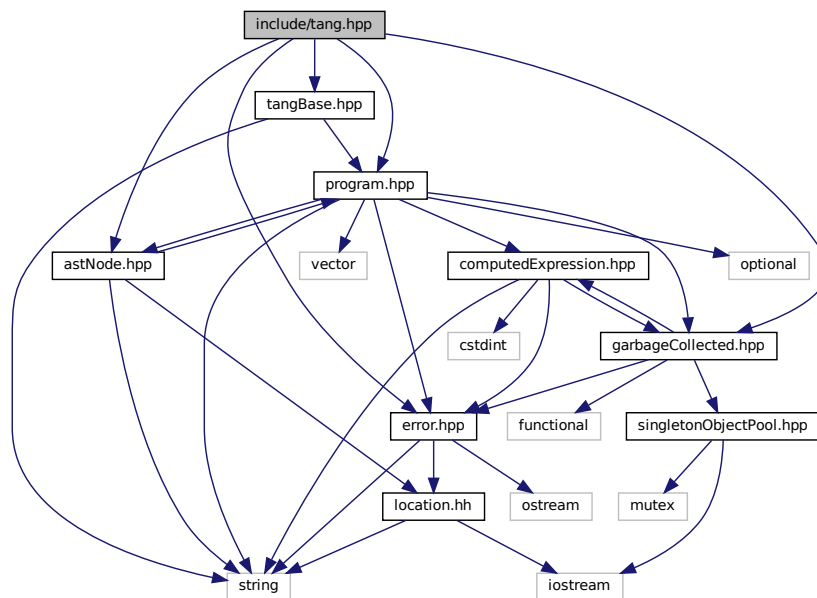
The threshold size to use when allocating blocks of data, measured in the number of instances of the object type.

6.21 include/tang.hpp File Reference

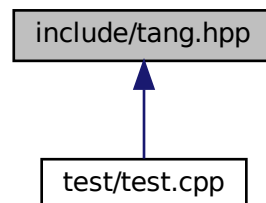
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

```
#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:



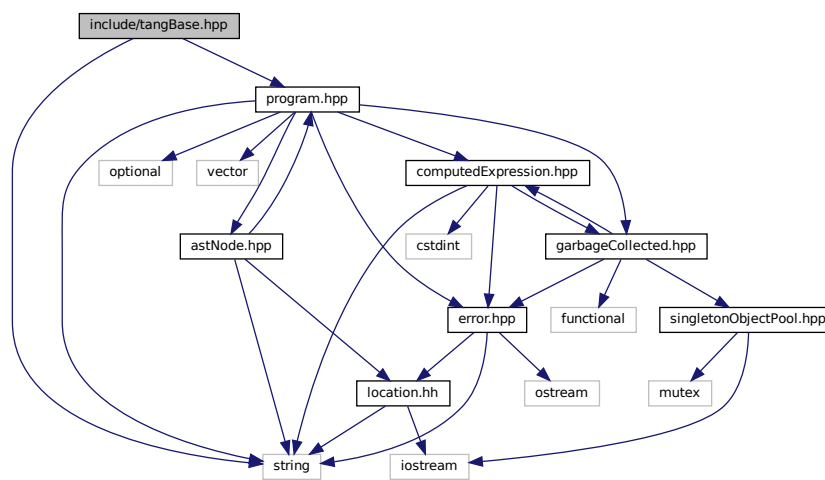
6.21.1 Detailed Description

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

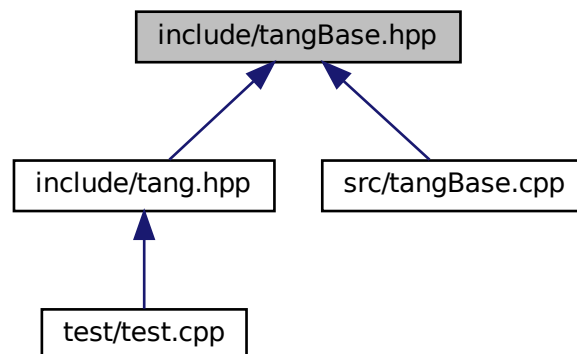
6.22 include/tangBase.hpp File Reference

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.

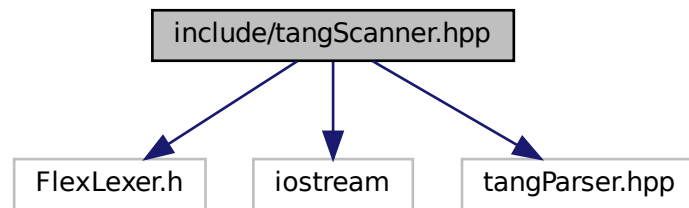
6.22.1 Detailed Description

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

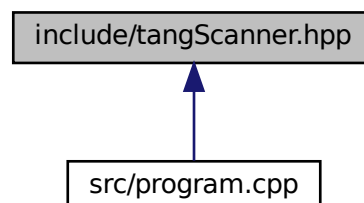
6.23 include/tangScanner.hpp File Reference

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

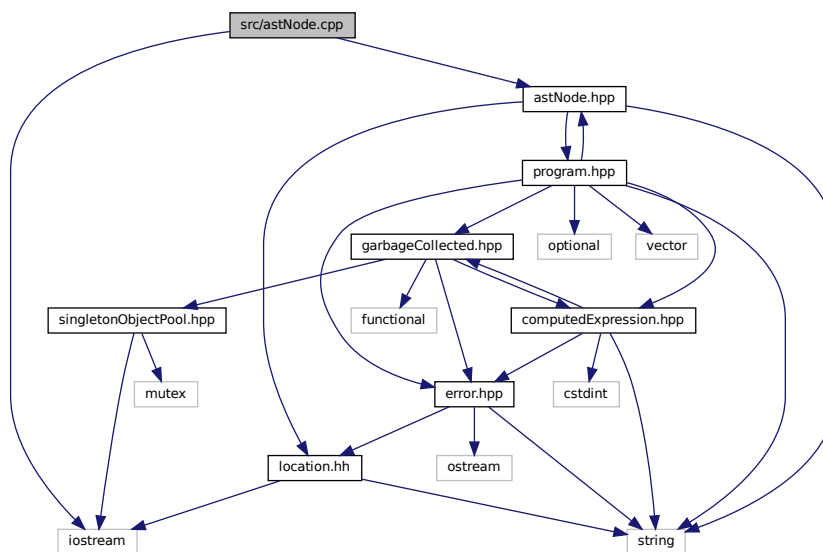
- `#define yyFlexLexer TangTangFlexLexer`
- `#define YY_DECL Tang::TangParser::symbol_type Tang::TangScanner::get_next_token\(\)`

6.23.1 Detailed Description

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

6.24 src/astNode.cpp File Reference

```
#include <iostream>
#include "astNode.hpp"
Include dependency graph for astNode.cpp:
```

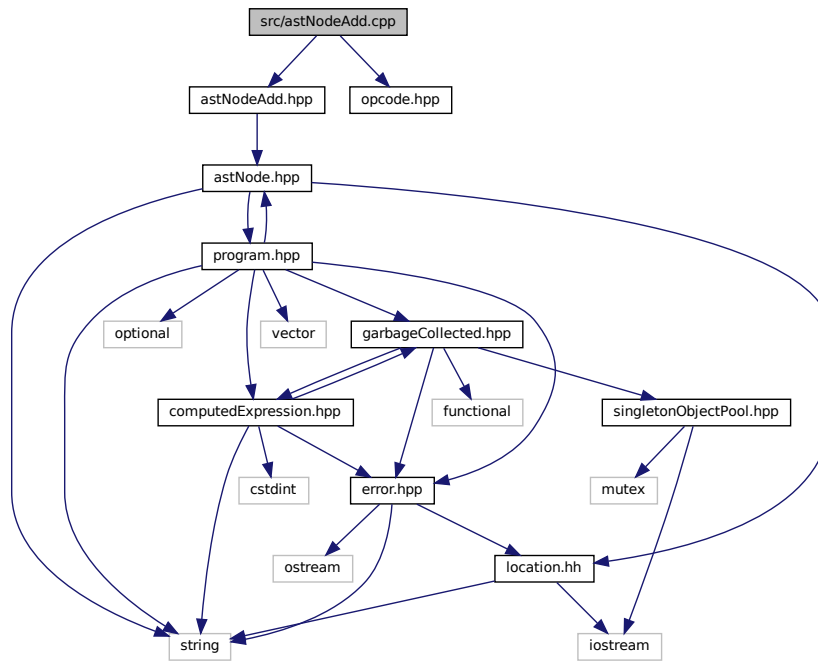


6.25 src/astNodeAdd.cpp File Reference

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

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

Include dependency graph for astNodeAdd.cpp:

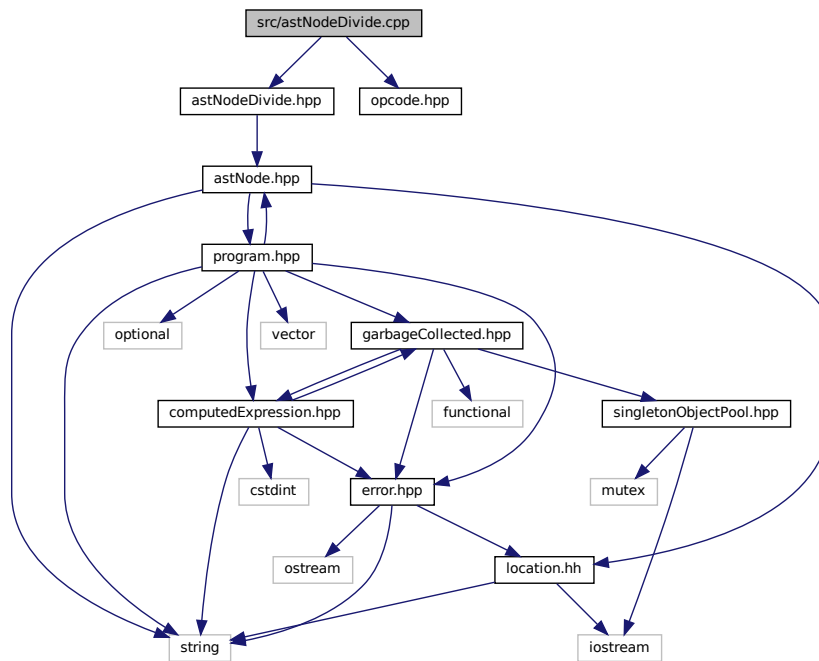


6.26 src/astNodeDivide.cpp File Reference

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

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

Include dependency graph for astNodeDivide.cpp:



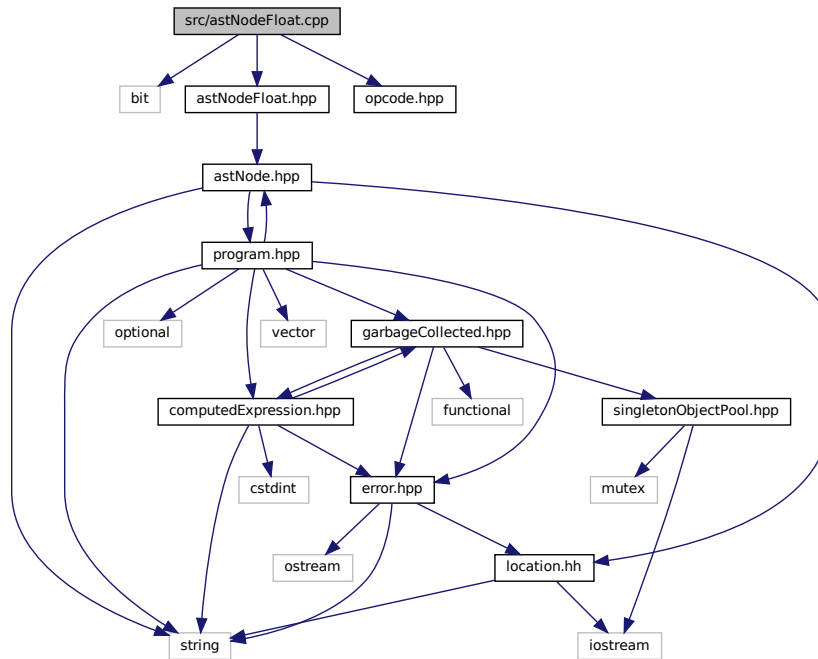
6.27 src/astNodeFloat.cpp File Reference

```

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

```

Include dependency graph for astNodeFloat.cpp:



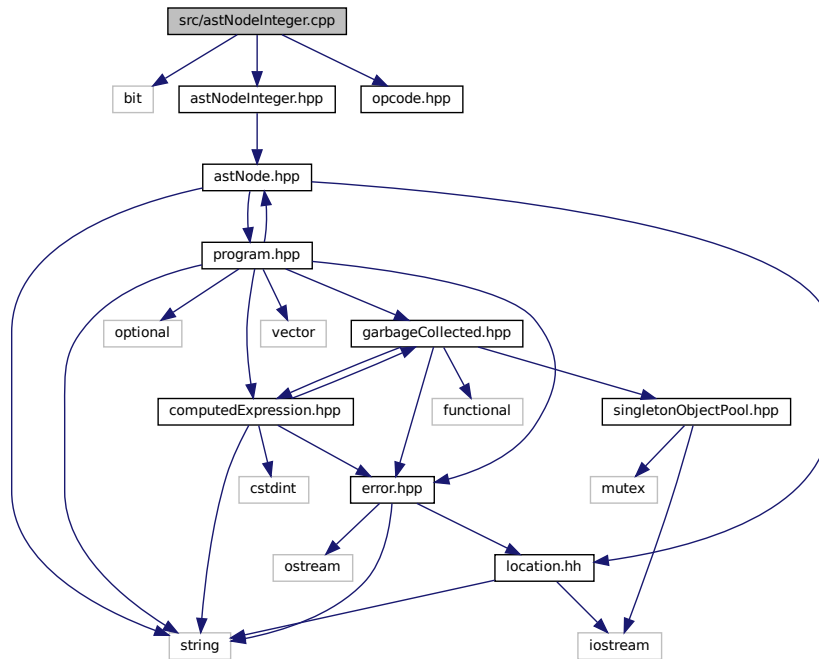
6.28 src/astNodeInteger.cpp File Reference

```

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

```

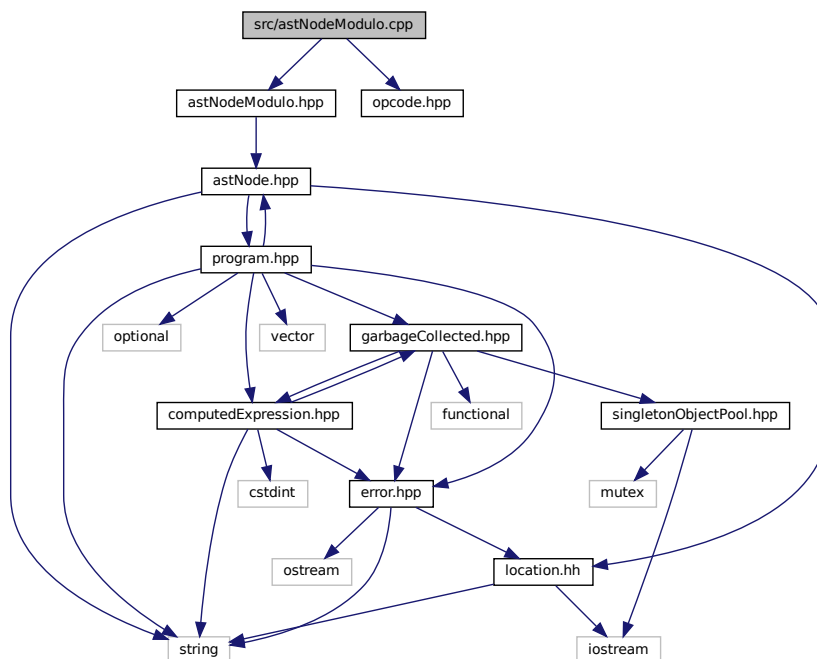
Include dependency graph for astNodeInteger.cpp:



6.29 src/astNodeModulo.cpp File Reference

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

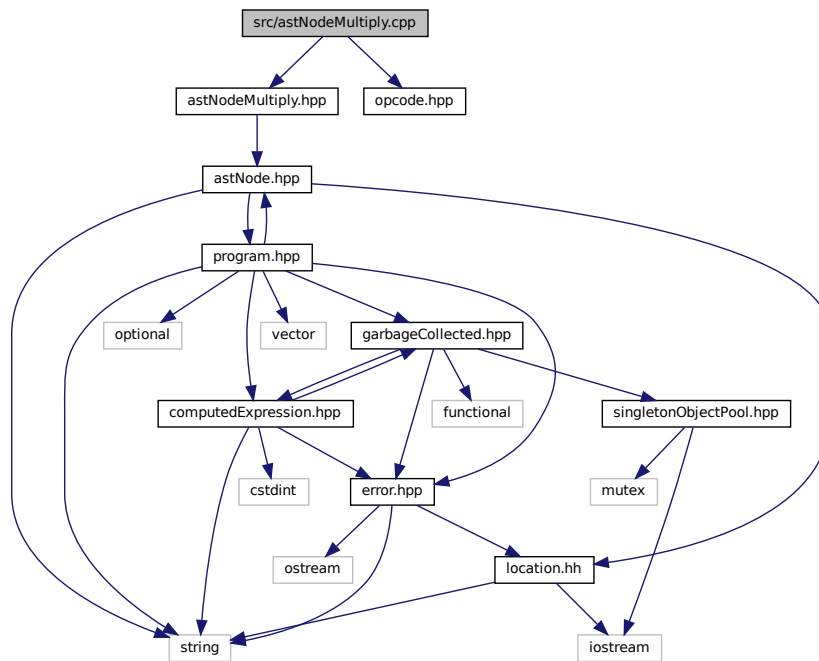

Include dependency graph for astNodeModulo.cpp:



6.30 src/astNodeMultiply.cpp File Reference

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

Include dependency graph for astNodeMultiply.cpp:



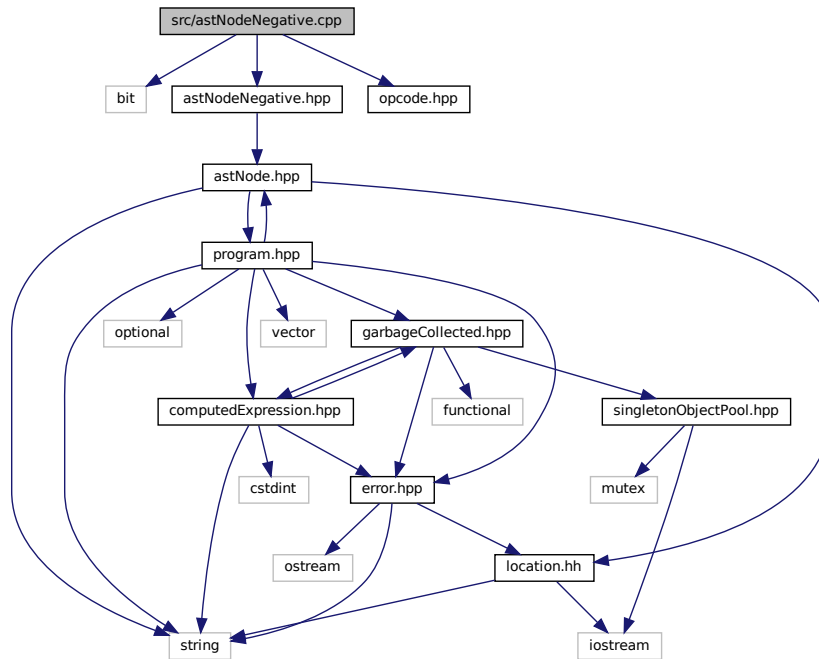
6.31 src/astNodeNegative.cpp File Reference

```

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

```

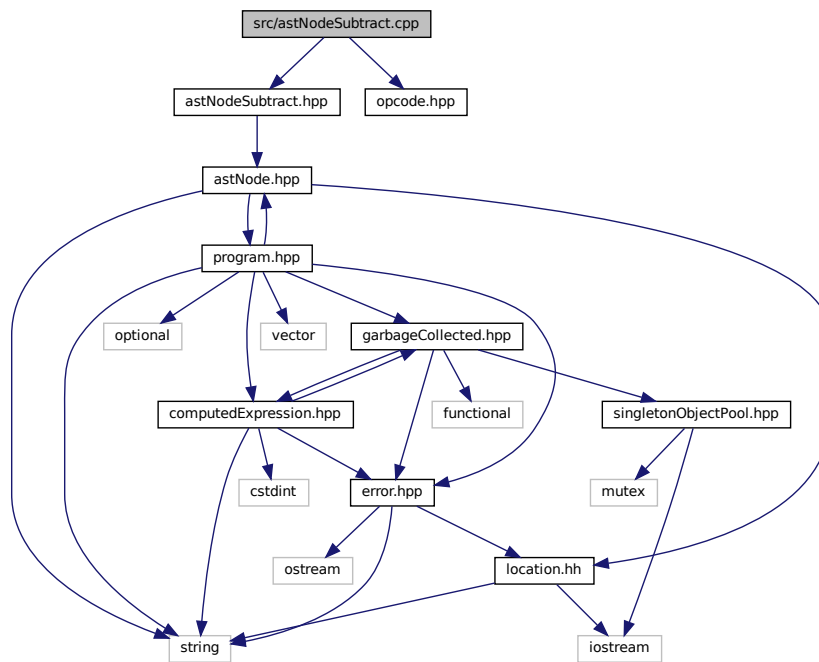
Include dependency graph for astNodeNegative.cpp:



6.32 src/astNodeSubtract.cpp File Reference

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

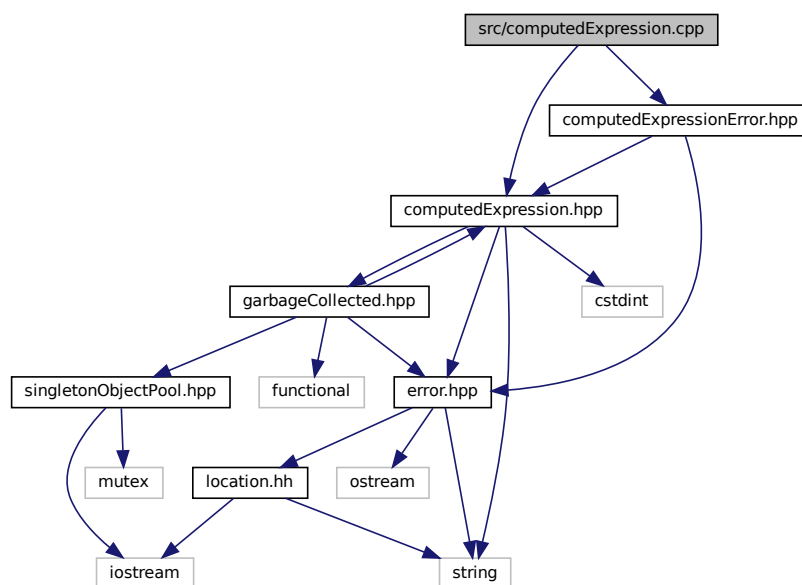
Include dependency graph for astNodeSubtract.cpp:



6.33 src/computedExpression.cpp File Reference

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

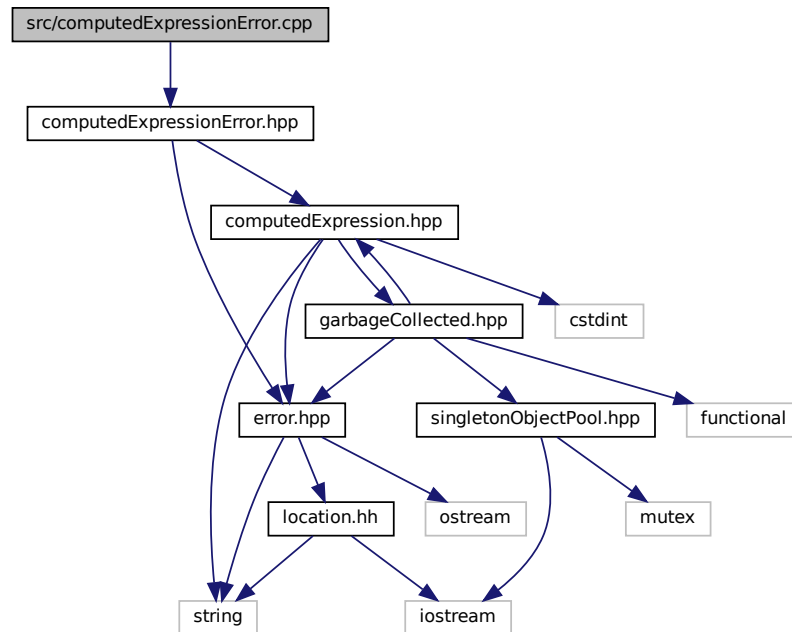
Include dependency graph for computedExpression.cpp:



6.34 src/computedExpressionError.cpp File Reference

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

Include dependency graph for computedExpressionError.cpp:



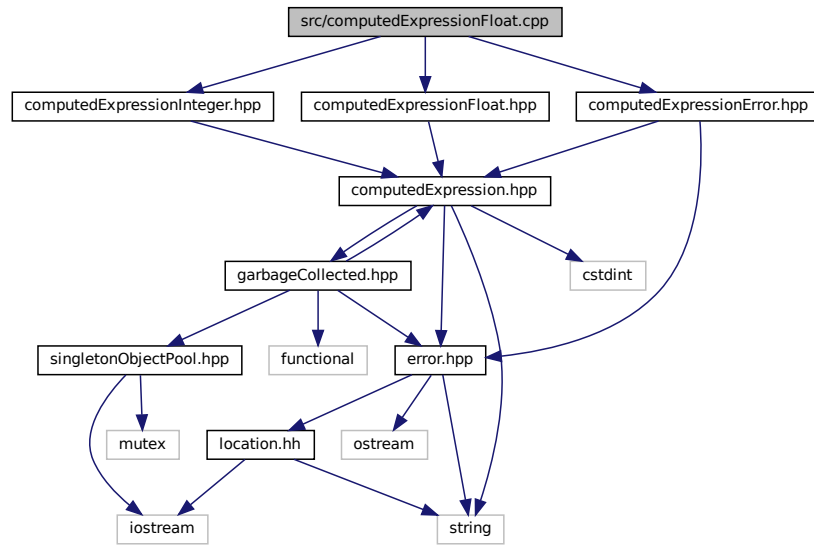
6.35 src/computedExpressionFloat.cpp File Reference

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

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

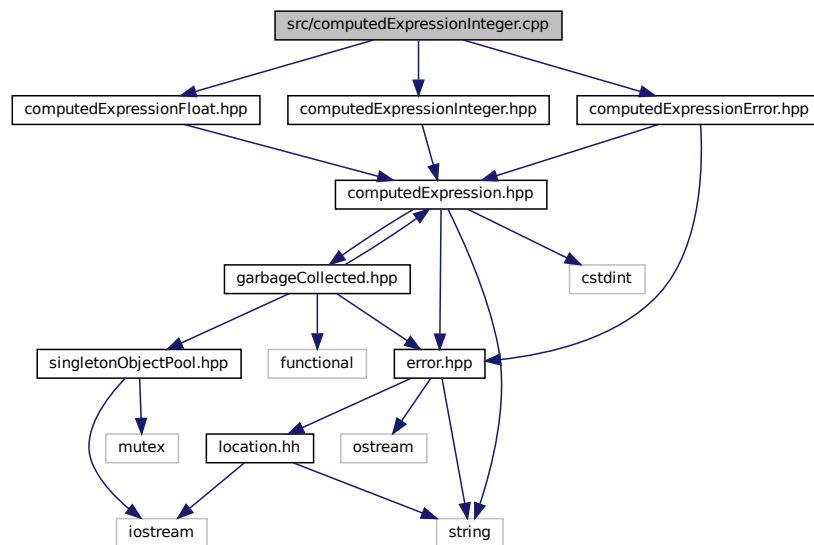
```
#include "computedExpressionError.hpp"
```

Include dependency graph for `computedExpressionFloat.cpp`:



6.36 src/computedExpressionInteger.cpp File Reference

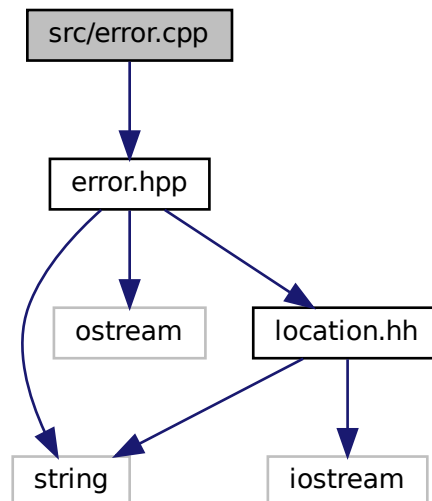
```
#include "computedExpressionInteger.hpp"
#include "computedExpressionFloat.hpp"
#include "computedExpressionError.hpp"
Include dependency graph for computedExpressionInteger.cpp:
```



6.37 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)`

6.37.1 Function Documentation

6.37.1.1 operator<<()

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

Parameters

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

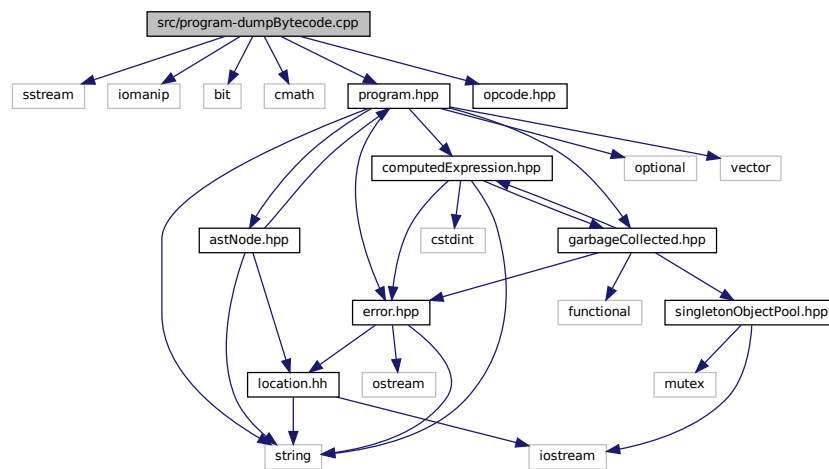
Returns

The output stream.

6.38 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.

6.38.1 Macro Definition Documentation

6.38.1.1 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.

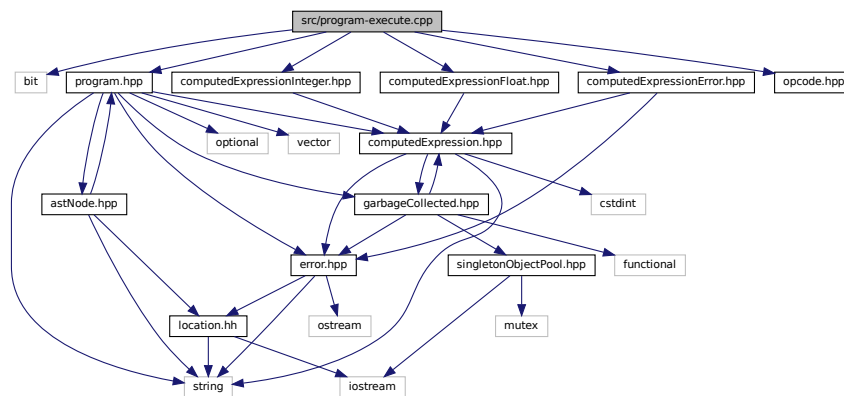
Parameters

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

6.39 src/program-execute.cpp File Reference

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

Include dependency graph for program-execute.cpp:



Macros

- `#define EXECUTEPROGRAMCHECK(x)`
Verify the size of the Bytecode vector so that it may be safely accessed.
- `#define STACKCHECK(x)`
Verify the size of the stack vector so that it may be safely accessed.

6.39.1 Macro Definition Documentation

6.39.1.1 EXECUTEPROGRAMCHECK

```
#define EXECUTEPROGRAMCHECK (
    x )
```

Value:

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

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

Parameters

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

6.39.1.2 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; \
}
```

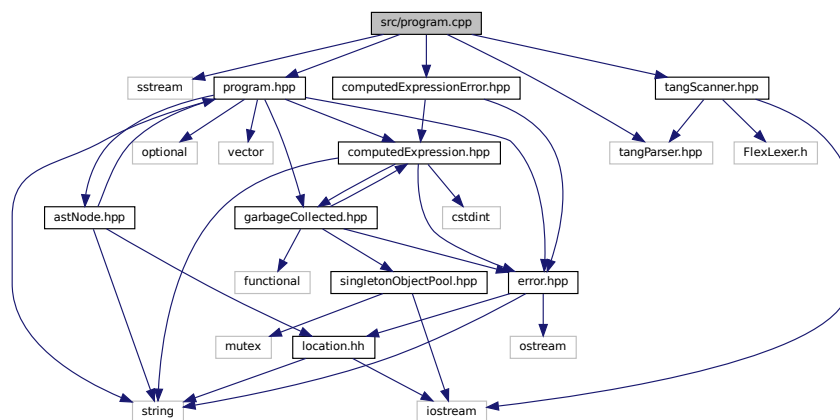
Verify the size of the stack vector so that it may be safely accessed.

Parameters

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

6.40 src/program.cpp File Reference

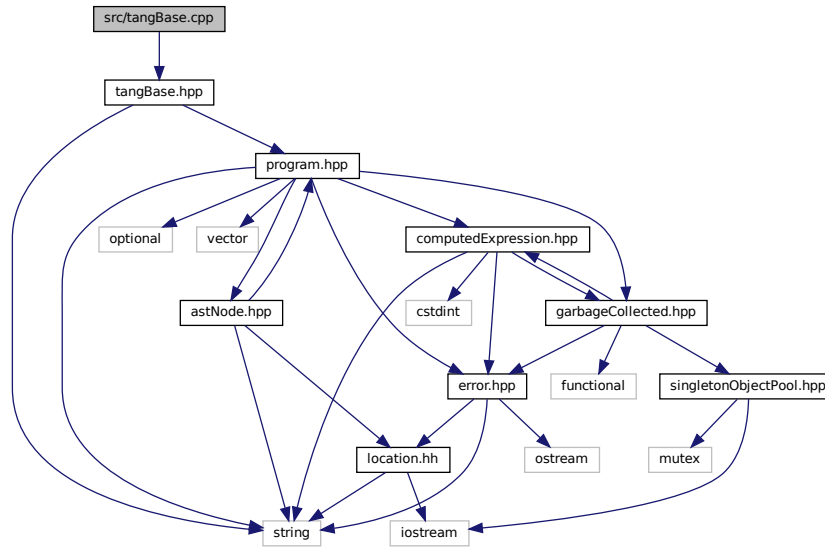
```
#include <sstream>
#include "program.hpp"
#include "tangScanner.hpp"
#include "tangParser.hpp"
#include "computedExpressionError.hpp"
Include dependency graph for program.cpp:
```



6.41 src/tangBase.cpp File Reference

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

Include dependency graph for tangBase.cpp:



6.42 test/test.cpp File Reference

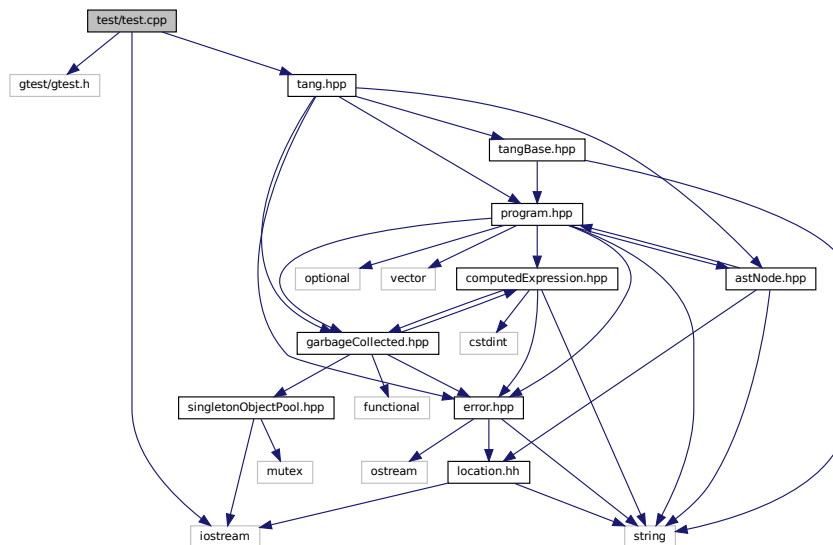
Test the general language behaviors.

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

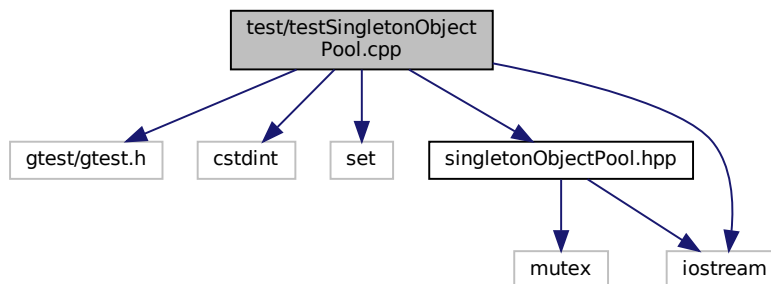
6.42.1 Detailed Description

Test the general language behaviors.

6.43 test/testSingletonObjectPool.cpp File Reference

```
#include <gtest/gtest.h>
#include <cstdint>
#include <set>
#include "singletonObjectPool.hpp"
#include <iostream>
```

Include dependency graph for testSingletonObjectPool.cpp:



Functions

- **TEST** (Singleton, SameForSameType)
- **TEST** (Singleton, DifferentForDifferentTypes)
- **TEST** (Get, SuccessiveCallsProduceDifferentMemoryAddresses)
- **TEST** (Recycle, RecycledObjectsIsReused)
- **TEST** (Get, SuccessiveCallsAreSequential)
- **TEST** (Get, KeepsGeneratingDifferentPointers)
- **TEST** (Recycle, WorksAfterLargeNumberOfAllocations)
- **int main** (int argc, char **argv)

Index

- __add
 - Tang::ComputedExpression, [37](#)
 - Tang::ComputedExpressionError, [42](#)
 - Tang::ComputedExpressionFloat, [48](#)
 - Tang::ComputedExpressionInteger, [54](#)
 - __divide
 - Tang::ComputedExpression, [37](#)
 - Tang::ComputedExpressionError, [43](#)
 - Tang::ComputedExpressionFloat, [49](#)
 - Tang::ComputedExpressionInteger, [55](#)
 - __modulo
 - Tang::ComputedExpression, [37](#)
 - Tang::ComputedExpressionError, [43](#)
 - Tang::ComputedExpressionFloat, [49](#)
 - Tang::ComputedExpressionInteger, [55](#)
 - __multiply
 - Tang::ComputedExpression, [38](#)
 - Tang::ComputedExpressionError, [43](#)
 - Tang::ComputedExpressionFloat, [50](#)
 - Tang::ComputedExpressionInteger, [56](#)
 - __negative
 - Tang::ComputedExpression, [38](#)
 - Tang::ComputedExpressionError, [44](#)
 - Tang::ComputedExpressionFloat, [50](#)
 - Tang::ComputedExpressionInteger, [56](#)
 - __subtract
 - Tang::ComputedExpression, [38](#)
 - Tang::ComputedExpressionError, [44](#)
 - Tang::ComputedExpressionFloat, [50](#)
 - Tang::ComputedExpressionInteger, [56](#)
- ~GarbageCollected
 - Tang::GarbageCollected, [64](#)
- ADD
 - opcode.hpp, [101](#)
- addBytecode
 - Tang::Program, [76](#)
- AstNode
 - Tang::AstNode, [11](#)
- AstNodeAdd
 - Tang::AstNodeAdd, [14](#)
- AstNodeDivide
 - Tang::AstNodeDivide, [17](#)
- AstNodeFloat
 - Tang::AstNodeFloat, [20](#)
- AstNodeInteger
 - Tang::AstNodeInteger, [23](#)
- AstNodeModulo
 - Tang::AstNodeModulo, [26](#)
- AstNodeMultiply
 - Tang::AstNodeMultiply, [29](#)
- AstNodeNegative
 - Tang::AstNodeNegative, [32](#)
- AstNodeSubtract
 - Tang::AstNodeSubtract, [35](#)
- build/generated/location.hh, [83](#)
- CodeType
 - Tang::Program, [76](#)
- compileScript
 - Tang::TangBase, [80](#)
- ComputedExpressionError
 - Tang::ComputedExpressionError, [42](#)
- ComputedExpressionFloat
 - Tang::ComputedExpressionFloat, [48](#)
- ComputedExpressionInteger
 - Tang::ComputedExpressionInteger, [54](#)
- DIVIDE
 - opcode.hpp, [101](#)
- dump
 - Tang::ComputedExpression, [39](#)
 - Tang::ComputedExpressionError, [44](#)
 - Tang::ComputedExpressionFloat, [51](#)
 - Tang::ComputedExpressionInteger, [57](#)
- dumpBytecode
 - Tang::Program, [77](#)
- DUMPPROGRAMCHECK
 - program-dumpBytecode.cpp, [118](#)
- Error
 - Tang::Error, [60](#)
- error.cpp
 - operator<<, [117](#)
- execute
 - Tang::Program, [77](#)
- EXECUTEPROGRAMCHECK
 - program-execute.cpp, [119](#)
- FLOAT
 - opcode.hpp, [101](#)
- GarbageCollected
 - Tang::GarbageCollected, [63](#), [64](#)
- get
 - Tang::SingletonObjectPool< T >, [78](#)
- get_next_token
 - Tang::TangScanner, [82](#)
- getAst
 - Tang::Program, [77](#)

- getCode
 - Tang::Program, 77
- getInstance
 - Tang::SingletonObjectPool< T >, 79
- getResult
 - Tang::Program, 78
- include/astNode.hpp, 85
- include/astNodeAdd.hpp, 86
- include/astNodeDivide.hpp, 87
- include/astNodeFloat.hpp, 88
- include/astNodeInteger.hpp, 89
- include/astNodeModulo.hpp, 90
- include/astNodeMultiply.hpp, 91
- include/astNodeNegative.hpp, 92
- include/astNodeSubtract.hpp, 93
- include/computedExpression.hpp, 94
- include/computedExpressionError.hpp, 95
- include/computedExpressionFloat.hpp, 96
- include/computedExpressionInteger.hpp, 97
- include/error.hpp, 97
- include/garbageCollected.hpp, 98
- include/macros.hpp, 99
- include/opcode.hpp, 100
- include/program.hpp, 101
- include/singletonObjectPool.hpp, 102
- include/tang.hpp, 103
- include/tangBase.hpp, 104
- include/tangScanner.hpp, 105
- INTEGER
 - opcode.hpp, 101
- is_equal
 - Tang::ComputedExpression, 39, 40
 - Tang::ComputedExpressionError, 45
 - Tang::ComputedExpressionFloat, 51, 52
 - Tang::ComputedExpressionInteger, 57, 58
- location.hh
 - operator<<, 84, 85
- macros.hpp
 - TANG_UNUSED, 100
- make
 - Tang::GarbageCollected, 64
- makeCopy
 - Tang::AstNode, 11
 - Tang::AstNodeAdd, 14
 - Tang::AstNodeDivide, 17
 - Tang::AstNodeFloat, 20
 - Tang::AstNodeInteger, 23
 - Tang::AstNodeModulo, 26
 - Tang::AstNodeMultiply, 29
 - Tang::AstNodeNegative, 32
 - Tang::AstNodeSubtract, 35
 - Tang::ComputedExpression, 40
 - Tang::ComputedExpressionError, 46
 - Tang::ComputedExpressionFloat, 52
 - Tang::ComputedExpressionInteger, 58
- MODULO
 - opcode.hpp, 101
- MULTIPLY
 - opcode.hpp, 101
- NEGATIVE
 - opcode.hpp, 101
- Opcode
 - opcode.hpp, 100
- opcode.hpp
 - ADD, 101
 - DIVIDE, 101
 - FLOAT, 101
 - INTEGER, 101
 - MODULO, 101
 - MULTIPLY, 101
 - NEGATIVE, 101
 - Opcode, 100
 - SUBTRACT, 101
- operator<<
 - error.cpp, 117
 - location.hh, 84, 85
 - Tang::Error, 61
 - Tang::GarbageCollected, 71
- operator*
 - Tang::GarbageCollected, 65
- operator+
 - Tang::GarbageCollected, 66
- operator-
 - Tang::GarbageCollected, 67
- operator->
 - Tang::GarbageCollected, 68
- operator/
 - Tang::GarbageCollected, 68
- operator=
 - Tang::GarbageCollected, 69
- operator==
 - Tang::GarbageCollected, 70
- operator%
 - Tang::GarbageCollected, 65
- Program
 - Tang::Program, 76
- program-dumpBytecode.cpp
 - DUMPPROGRAMCHECK, 118
- program-execute.cpp
 - EXECUTEPROGRAMCHECK, 119
 - STACKCHECK, 120
- recycle
 - Tang::SingletonObjectPool< T >, 79
- Script
 - Tang::Program, 76
- src/astNode.cpp, 106
- src/astNodeAdd.cpp, 107
- src/astNodeDivide.cpp, 107
- src/astNodeFloat.cpp, 108
- src/astNodeInteger.cpp, 109

- src/astNodeModulo.cpp, 110
- src/astNodeMultiply.cpp, 111
- src/astNodeNegative.cpp, 112
- src/astNodeSubtract.cpp, 113
- src/computedExpression.cpp, 114
- src/computedExpressionError.cpp, 115
- src/computedExpressionFloat.cpp, 115
- src/computedExpressionInteger.cpp, 116
- src/error.cpp, 117
- src/program-dumpBytecode.cpp, 118
- src/program-execute.cpp, 119
- src/program.cpp, 120
- src/tangBase.cpp, 121
- STACKCHECK
 - program-execute.cpp, 120
- SUBTRACT
 - opcode.hpp, 101
- Tang::AstNode, 9
 - AstNode, 11
 - makeCopy, 11
- Tang::AstNodeAdd, 12
 - AstNodeAdd, 14
 - makeCopy, 14
- Tang::AstNodeDivide, 15
 - AstNodeDivide, 17
 - makeCopy, 17
- Tang::AstNodeFloat, 18
 - AstNodeFloat, 20
 - makeCopy, 20
- Tang::AstNodeInteger, 21
 - AstNodeInteger, 23
 - makeCopy, 23
- Tang::AstNodeModulo, 24
 - AstNodeModulo, 26
 - makeCopy, 26
- Tang::AstNodeMultiply, 27
 - AstNodeMultiply, 29
 - makeCopy, 29
- Tang::AstNodeNegative, 30
 - AstNodeNegative, 32
 - makeCopy, 32
- Tang::AstNodeSubtract, 33
 - AstNodeSubtract, 35
 - makeCopy, 35
- Tang::ComputedExpression, 36
 - __add, 37
 - __divide, 37
 - __modulo, 37
 - __multiply, 38
 - __negative, 38
 - __subtract, 38
 - dump, 39
 - is_equal, 39, 40
 - makeCopy, 40
- Tang::ComputedExpressionError, 41
 - __add, 42
 - __divide, 43
 - __modulo, 43
 - __multiply, 43
 - __negative, 44
 - __subtract, 44
- ComputedExpressionError, 42
- dump, 44
- is_equal, 45
- makeCopy, 46
- Tang::ComputedExpressionFloat, 47
 - __add, 48
 - __divide, 49
 - __modulo, 49
 - __multiply, 50
 - __negative, 50
 - __subtract, 50
- ComputedExpressionFloat, 48
- dump, 51
- is_equal, 51, 52
- makeCopy, 52
- Tang::ComputedExpressionInteger, 53
 - __add, 54
 - __divide, 55
 - __modulo, 55
 - __multiply, 56
 - __negative, 56
 - __subtract, 56
- ComputedExpressionInteger, 54
- dump, 57
- is_equal, 57, 58
- makeCopy, 58
- Tang::Error, 59
 - Error, 60
 - operator<<, 61
- Tang::GarbageCollected, 61
 - ~GarbageCollected, 64
 - GarbageCollected, 63, 64
 - make, 64
 - operator<<, 71
 - operator*, 65
 - operator+, 66
 - operator-, 67
 - operator->, 68
 - operator/, 68
 - operator=, 69
 - operator==, 70
 - operator%, 65
- Tang::location, 71
- Tang::position, 73
- Tang::Program, 74
 - addBytecode, 76
 - CodeType, 76
 - dumpBytecode, 77
 - execute, 77
 - getAst, 77
 - getCode, 77
 - getResult, 78
 - Program, 76
 - Script, 76
 - Template, 76

- Tang::SingletonObjectPool< T >, [78](#)
 - get, [78](#)
 - getInstance, [79](#)
 - recycle, [79](#)
- Tang::TangBase, [79](#)
 - compileScript, [80](#)
 - TangBase, [80](#)
- Tang::TangScanner, [81](#)
 - get_next_token, [82](#)
 - TangScanner, [82](#)
- TANG_UNUSED
 - macros.hpp, [100](#)
- TangBase
 - Tang::TangBase, [80](#)
- TangScanner
 - Tang::TangScanner, [82](#)
- Template
 - Tang::Program, [76](#)
- test/test.cpp, [121](#)
- test/testSingletonObjectPool.cpp, [122](#)