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::AstNodeFloat Class Reference . . . 15
5.3.1 Detailed Description . . . 17
5.3.2 Constructor & Destructor Documentation . . . 17
5.3.2.1 AstNodeFloat() . . . 17
5.3.3 Member Function Documentation . . . 17
5.3.3.1 makeCopy() . . . 17
5.4 Tang::AstNodeInteger Class Reference . . . 18
5.4.1 Detailed Description . . . 20
5.4.2 Constructor & Destructor Documentation . . . 20
5.4.2.1 AstNodeInteger() . . . 20
5.4.3 Member Function Documentation . . . 20
5.4.3.1 makeCopy() . . . 20
5.5 Tang::ComputedExpression Class Reference . . . 21
5.5.1 Detailed Description . . . 21
5.5.2 Member Function Documentation . . . 21
5.5.2.1 __add() . . . 21
5.5.2.2 dump() . . . 22

5.5.2.3 is_equal() [1/2] . . . 22
5.5.2.4 is_equal() [2/2] . . . 22
5.5.2.5 makeCopy() . . . 23
5.6 Tang::ComputedExpressionFloat Class Reference . . . 23
5.6.1 Detailed Description . . . 24
5.6.2 Constructor & Destructor Documentation . . . 24
5.6.2.1 ComputedExpressionFloat() . . . 24
5.6.3 Member Function Documentation . . . 25
5.6.3.1 __add() . . . 25
5.6.3.2 dump() . . . 25
5.6.3.3 is_equal() [1/2] . . . 25
5.6.3.4 is_equal() [2/2] . . . 26
5.6.3.5 makeCopy() . . . 26
5.7 Tang::ComputedExpressionInteger Class Reference . . . 27
5.7.1 Detailed Description . . . 28
5.7.2 Constructor & Destructor Documentation . . . 28
5.7.2.1 ComputedExpressionInteger() . . . 28
5.7.3 Member Function Documentation . . . 28
5.7.3.1 __add() . . . 28
5.7.3.2 dump() . . . 29
5.7.3.3 is_equal() [1/2] . . . 29
5.7.3.4 is_equal() [2/2] . . . 29
5.7.3.5 makeCopy() . . . 30
5.8 Tang::Error Class Reference . . . 31
5.8.1 Detailed Description . . . 32
5.8.2 Constructor & Destructor Documentation . . . 32
5.8.2.1 Error() . . . 32
5.9 Tang::GarbageCollected Class Reference . . . 32
5.9.1 Detailed Description . . . 34
5.9.2 Constructor & Destructor Documentation . . . 34
5.9.2.1 GarbageCollected() [1/3] . . . 34
5.9.2.2 GarbageCollected() [2/3] . . . 34
5.9.2.3 ~GarbageCollected() . . . 34
5.9.2.4 GarbageCollected() [3/3] . . . 35
5.9.3 Member Function Documentation . . . 36
5.9.3.1 make() . . . 36
5.9.3.2 operator*() . . . 36
5.9.3.3 operator->() . . . 37
5.9.3.4 operator=() [1/2] . . . 37
5.9.3.5 operator=() [2/2] . . . 37
5.9.3.6 operator==() [1/2] . . . 38
5.9.3.7 operator==() [2/2] . . . 38

5.9.4 Friends And Related Function Documentation . . . 38
5.9.4.1 operator<< . . . 39
5.10 Tang::location Class Reference . . . 39
5.10.1 Detailed Description . . . 40
5.11 Tang::position Class Reference . . . 41
5.11.1 Detailed Description . . . 42
5.12 Tang::Program Class Reference . . . 42
5.12.1 Detailed Description . . . 43
5.12.2 Member Enumeration Documentation . . . 43
5.12.2.1 CodeType . . . 43
5.12.3 Constructor & Destructor Documentation . . . 44
5.12.3.1 Program() . . . 44
5.12.4 Member Function Documentation . . . 44
5.12.4.1 addBytecode() . . . 44
5.12.4.2 dumpBytecode() . . . 44
5.12.4.3 execute() . . . 45
5.12.4.4 getAst() . . . 45
5.12.4.5 getCode() . . . 45
5.12.4.6 getResult() . . . 46
5.13 Tang::SingletonObjectPool< T > Class Template Reference . . . 46
5.13.1 Member Function Documentation . . . 46
5.13.1.1 get() . . . 46
5.13.1.2 getInstance() . . . 47
5.13.1.3 recycle() . . . 47
5.14 Tang::TangBase Class Reference . . . 47
5.14.1 Detailed Description . . . 48
5.14.2 Constructor & Destructor Documentation . . . 48
5.14.2.1 TangBase() . . . 48
5.14.3 Member Function Documentation . . . 48
5.14.3.1 compileScript() . . . 48
5.15 Tang::TangScanner Class Reference . . . 49
5.15.1 Detailed Description . . . 49
5.15.2 Constructor & Destructor Documentation . . . 50
5.15.2.1 TangScanner() . . . 50
5.15.3 Member Function Documentation . . . 50
5.15.3.1 get_next_token() . . . 50

6 File Documentation **51**

6.1 build/generated/location.hh File Reference . . . 51
6.1.1 Detailed Description . . . 52
6.1.2 Function Documentation . . . 52
6.1.2.1 operator<<() [1/2] . . . 53

6.1.2.2 operator<<() [2/2] . . . 53
6.2 include/ast.hpp File Reference . . . 53
6.2.1 Detailed Description . . . 54
6.3 include/computedExpression.hpp File Reference . . . 55
6.4 include/error.hpp File Reference . . . 56
6.4.1 Detailed Description . . . 57
6.5 include/garbageCollected.hpp File Reference . . . 57
6.6 include/macros.hpp File Reference . . . 58
6.6.1 Detailed Description . . . 58
6.6.2 Macro Definition Documentation . . . 58
6.6.2.1 TANG_UNUSED . . . 58
6.7 include/opcode.hpp File Reference . . . 59
6.7.1 Detailed Description . . . 59
6.7.2 Enumeration Type Documentation . . . 59
6.7.2.1 Opcode . . . 59
6.8 include/program.hpp File Reference . . . 59
6.8.1 Detailed Description . . . 61
6.9 include/singletonObjectPool.hpp File Reference . . . 61
6.10 include/tang.hpp File Reference . . . 62
6.10.1 Detailed Description . . . 62
6.11 include/tangBase.hpp File Reference . . . 62
6.11.1 Detailed Description . . . 63
6.12 include/tangScanner.hpp File Reference . . . 64
6.12.1 Detailed Description . . . 65
6.13 src/ast.cpp File Reference . . . 65
6.14 src/computedExpression.cpp File Reference . . . 66
6.15 src/error.cpp File Reference . . . 67
6.16 src/program.cpp File Reference . . . 67
6.16.1 Macro Definition Documentation . . . 68
6.16.1.1 DUMPPROGRAMCHECK . . . 68
6.16.1.2 EXECUTEPROGRAMCHECK . . . 68
6.16.1.3 STACKCHECK . . . 69
6.17 src/tangBase.cpp File Reference . . . 69
6.18 test/testSingletonObjectPool.cpp File Reference . . . 69

Index **71**

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::AstNodeFloat 15
  Tang::AstNodeInteger 18
Tang::ComputedExpression 21
  Tang::ComputedExpressionFloat 23
  Tang::ComputedExpressionInteger 27
Tang::Error 31
Tang::GarbageCollected 32
Tang::location 39
Tang::position 41
Tang::Program 42
Tang::SingletonObjectPool< T > 46
Tang::TangBase 47
TangTangFlexLexer
  Tang::TangScanner 49

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::AstNodeFloat
: An AstNode that represents an float literal . . . 15

Tang::AstNodeInteger
: An AstNode that represents an integer literal . . . 18

Tang::ComputedExpression
: Represents the result of a computation that has been executed . . . 21

Tang::ComputedExpressionFloat
: Represents a Float that is the result of a computation . . . 23

Tang::ComputedExpressionInteger
: Represents an Integer that is the result of a computation . . . 27

Tang::Error
: Used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error . . . 31

Tang::GarbageCollected
: A container that acts as a resource-counting garbage collector for the specified type . . . 32

Tang::location
: Two points in a source file . . . 39

Tang::position
: A point in a source file . . . 41

Tang::Program
: Represents a compiled script or template that may be executed . . . 42

Tang::SingletonObjectPool< T > . . . 46

Tang::TangBase
: The base class for the Tang programming language . . . 47

Tang::TangScanner
: The Flex lexer class for the main Tang language . . . 49

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

build/generated/location.hh
    Define the Tang ::location class . . . 51
include/ast.hpp
    Define the Tang::AstNode and its associated/derivative classes . . . 53
include/computedExpression.hpp . . . 55
include/error.hpp
    Define the Tang::Error class used to describe syntax and runtime errors . . . 56
include/garbageCollected.hpp . . . 57
include/macros.hpp
    Contains generic macros . . . 58
include/opcode.hpp
    Declare the Opcodes used in the Bytecode representation of a program . . . 59
include/program.hpp
    Define the Tang::Program class used to compile and execute source code . . . 59
include/singletonObjectPool.hpp . . . 61
include/tang.hpp
    Header file supplied for use by 3rd party code so that they can easily include all necessary headers . . . 62
include/tangBase.hpp
    Defines the Tang::TangBase class used to interact with Tang . . . 62
include/tangScanner.hpp
    Defines the Tang::TangScanner used to tokenize a Tang script . . . 64
src/ast.cpp . . . 65
src/computedExpression.cpp . . . 66
src/error.cpp . . . 67
src/program.cpp . . . 67
src/tangBase.cpp . . . 69
test/testSingletonObjectPool.cpp . . . 69

Chapter 5

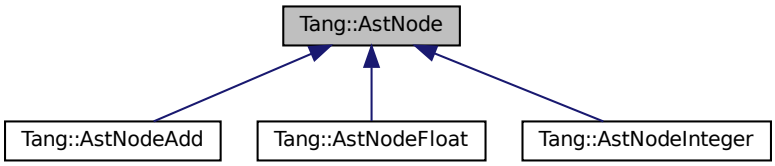
Class Documentation

5.1 Tang::AstNode Class Reference

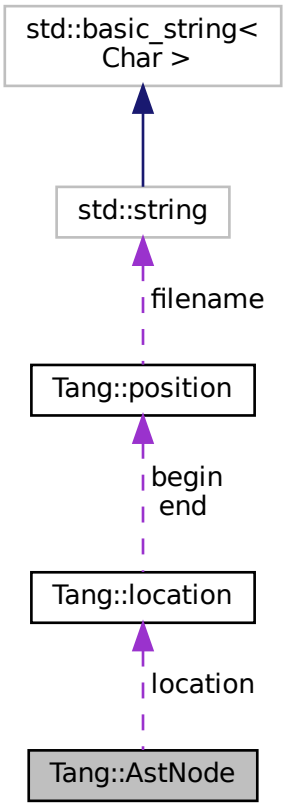
Base class for representing nodes of an Abstract Syntax Tree (AST).

`#include <ast.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

loc	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::AstNodeAdd, Tang::AstNodeFloat, and Tang::AstNodeInteger.

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

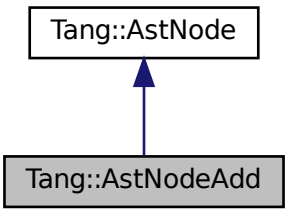
- include/ast.hpp
- src/ast.cpp

5.2 Tang::AstNodeAdd Class Reference

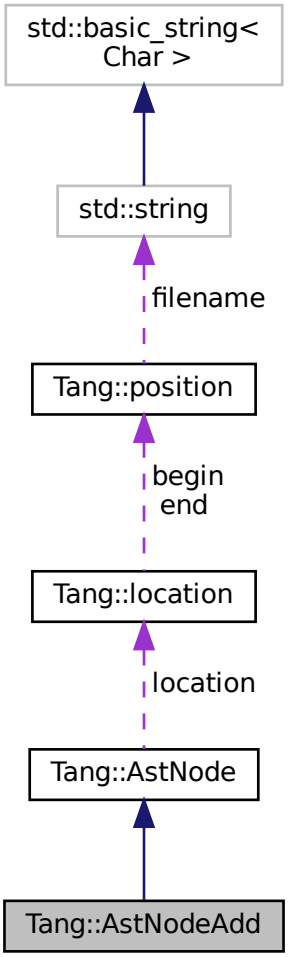
An AstNode that represents a "+" expression.

`#include <ast.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

lhs	The left hand side expression.
rhs	The right hand side expression.
loc	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:

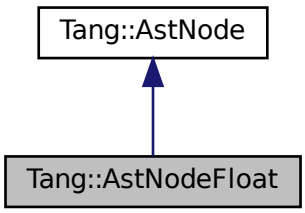
- include/ast.hpp
- src/ast.cpp

5.3 Tang::AstNodeFloat Class Reference

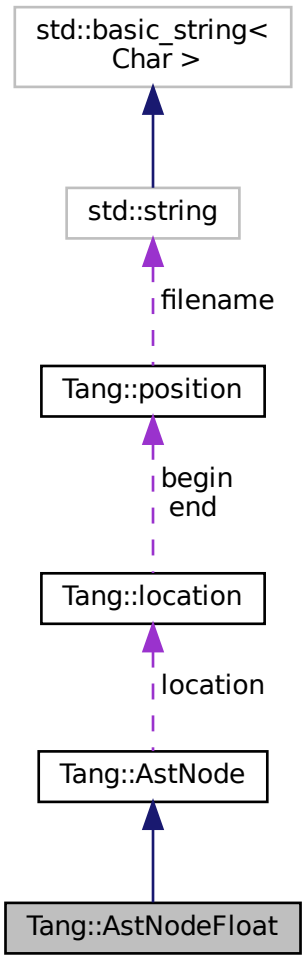
An AstNode that represents an float literal.

`#include <ast.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.

5.3.1 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.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 AstNodeFloat()

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

The constructor.

Parameters

number	The number to represent.
loc	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 * 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:

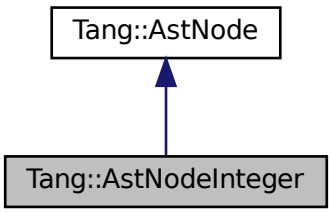
- include/ast.hpp
- src/ast.cpp

5.4 Tang::AstNodeInteger Class Reference

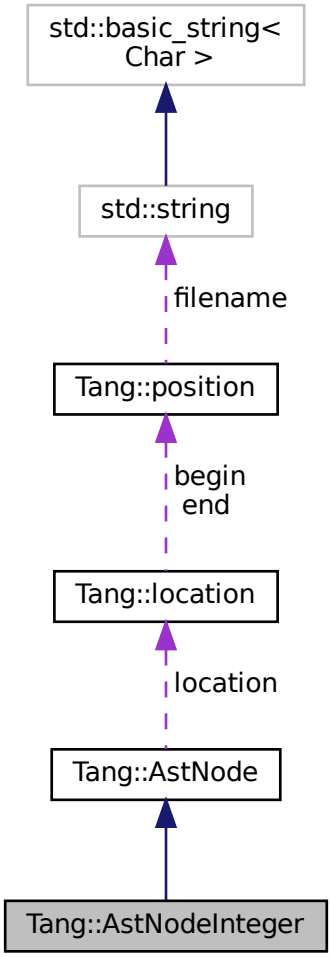
An AstNode that represents an integer literal.

`#include <ast.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.

5.4.1 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.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 AstNodeInteger()

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

The constructor.

Parameters

number	The number to represent.
loc	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 * 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:

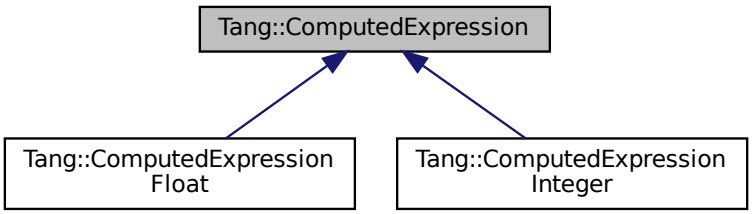
- include/ast.hpp
- src/ast.cpp

5.5 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 GarbageCollected __add (const GarbageCollected &rhs) const
  Compute the result of adding this value and the supplied value.

5.5.1 Detailed Description

Represents the result of a computation that has been executed.

5.5.2 Member Function Documentation

5.5.2.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

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

5.5.2.2 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::ComputedExpressionFloat, and Tang::ComputedExpressionInteger.

5.5.2.3 is_equal() [1/2]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.5.2.4 is_equal() [2/2]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.5.2.5 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::ComputedExpressionFloat, and Tang::ComputedExpressionInteger.

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

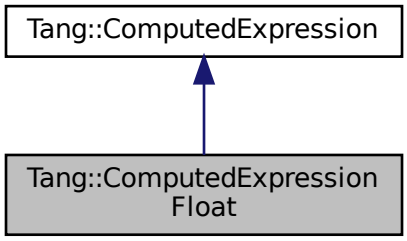
- include/computedExpression.hpp
- src/computedExpression.cpp

5.6 Tang::ComputedExpressionFloat Class Reference

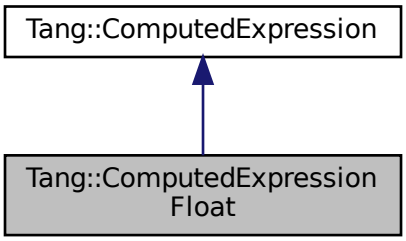
Represents a Float that is the result of a computation.

```
#include <computedExpression.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.

Friends

- class **ComputedExpressionInteger**

5.6.1 Detailed Description

Represents a Float that is the result of a computation.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 ComputedExpressionFloat()

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

Construct a Float result.

Parameters

val	The float value.

5.6.3 Member Function Documentation

5.6.3.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.6.3.2 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.

5.6.3.3 is_equal() [1/2]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.6.3.4 is_equal() [2/2]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.6.3.5 makeCopy()

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

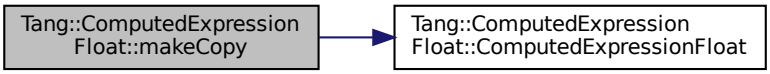
Make a copy of the ComputedExpression (recursively, if appropriate).

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:

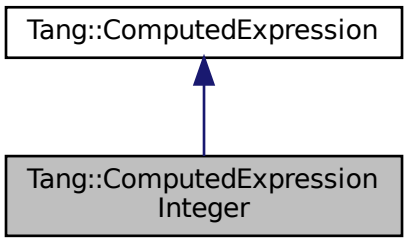
- include/computedExpression.hpp
- src/computedExpression.cpp

5.7 Tang::ComputedExpressionInteger Class Reference

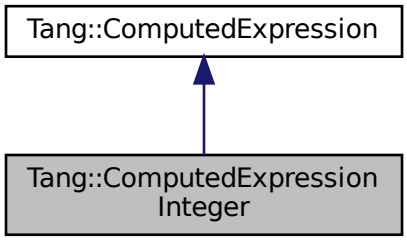
Represents an Integer that is the result of a computation.

`#include <computedExpression.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.

Friends

- class **ComputedExpressionFloat**

5.7.1 Detailed Description

Represents an Integer that is the result of a computation.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 ComputedExpressionInteger()

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

Construct an Integer result.

Parameters

val	The integer value.

5.7.3 Member Function Documentation

5.7.3.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.7.3.2 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.

5.7.3.3 is_equal() [1/2]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.7.3.4 is_equal() [2/2]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.7.3.5 makeCopy()

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

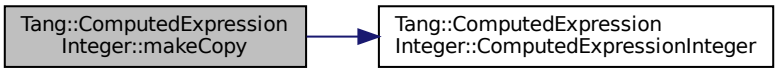
Make a copy of the ComputedExpression (recursively, if appropriate).

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:

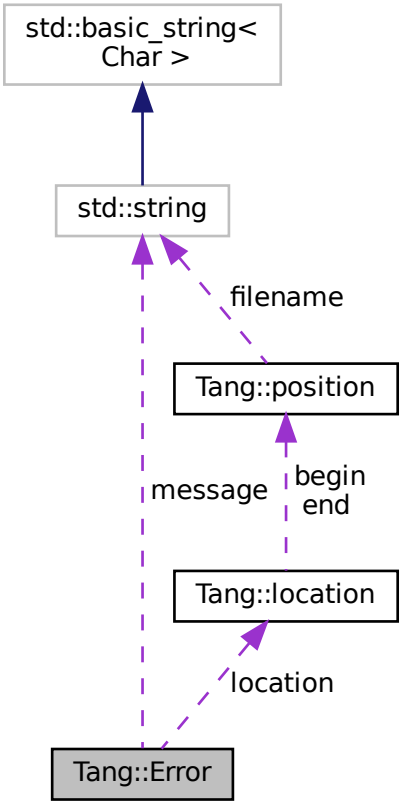
- include/computedExpression.hpp
- src/computedExpression.cpp

5.8 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, 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.

5.8.1 Detailed Description

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

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Error()

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

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

Parameters

message	The error message as a string.
location	The location of the error.

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

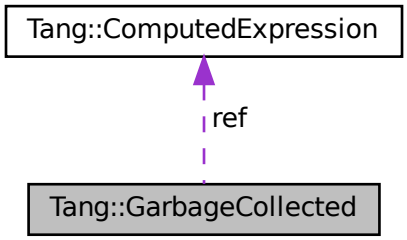
- include/error.hpp
- src/error.cpp

5.9 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.
- GarbageCollected **operator+** (const GarbageCollected &lhs) const

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.

5.9.1 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.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 GarbageCollected() [1/3]

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

Copy Constructor.

Parameters

The	other GarbageCollected object to copy.

5.9.2.2 GarbageCollected() [2/3]

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

Move Constructor.

Parameters

The	other GarbageCollected object to move.

5.9.2.3 ~GarbageCollected()

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

Destructor.

Clean up the tracked object, if appropriate.

5.9.2.4 GarbageCollected() [3/3]

```
Tang::GarbageCollected::GarbageCollected ( )  [inline], [protected]
```

Constructs a garbage-collected object of the specified type.

It is private so that a GarbageCollected object can only be created using the GarbageCollected::make() function.

Parameters

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

5.9.3 Member Function Documentation

5.9.3.1 make()

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

Creates a garbage-collected object of the specified type.

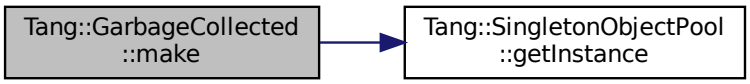
Parameters

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

Returns

A GarbageCollected object.

Here is the call graph for this function:



5.9.3.2 operator∗()

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

Access the tracked object.

Returns

A reference to the tracked object.

5.9.3.3 operator->()

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

Access the tracked object as a pointer.

Returns

A pointer to the tracked object.

5.9.3.4 operator=() [1/2]

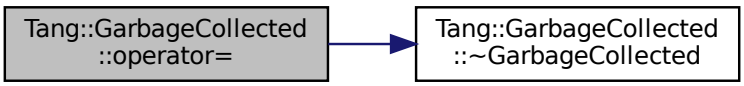
```
GarbageCollected& Tang::GarbageCollected::operator= (
            const GarbageCollected & other )  [inline]
```

Copy Assignment.

Parameters

The	other GarbageCollected object.

Here is the call graph for this function:



5.9.3.5 operator=() [2/2]

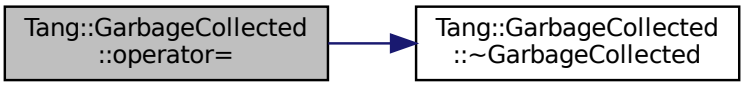
```
GarbageCollected& Tang::GarbageCollected::operator= (
            GarbageCollected && other )  [inline]
```

Move Assignment.

Parameters

The	other GarbageCollected object.

Here is the call graph for this function:



5.9.3.6 operator==() [1/2]

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

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.9.3.7 operator==() [2/2]

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

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.9.4 Friends And Related Function Documentation

5.9.4.1 operator<<

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

Add friendly output.

Parameters

out	The output stream.
gc	The GarbageCollected value.

Returns

The output stream.

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

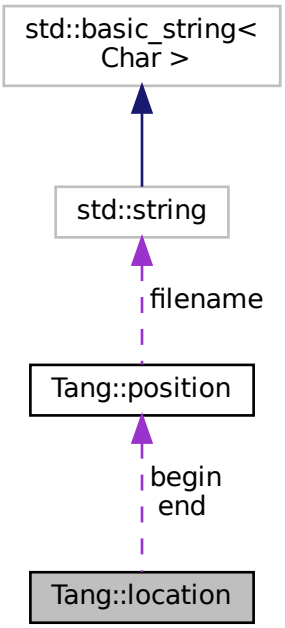
- include/garbageCollected.hpp
- src/garbageCollected.cpp

5.10 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.

5.10.1 Detailed Description

Two points in a source file.

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

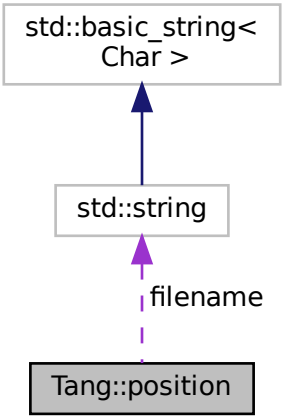
- build/generated/location.hh

5.11 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.

5.11.1 Detailed Description

A point in a source file.

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

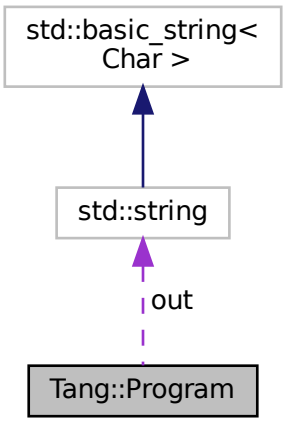
- build/generated/location.hh

5.12 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.

5.12.1 Detailed Description

Represents a compiled script or template that may be executed.

5.12.2 Member Enumeration Documentation

5.12.2.1 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.

5.12.3 Constructor & Destructor Documentation

5.12.3.1 Program()

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

Create a compiled program using the provided code.

Parameters

code	The code to be compiled.
codeType	Whether the code is a `Script` or `Template`.

5.12.4 Member Function Documentation

5.12.4.1 addBytecode()

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

Add a uint64_t to the Bytecode.

Parameters

op	The value to add to the Bytecode.

5.12.4.2 dumpBytecode()

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

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

Returns

A string containing the Opcode representation.

5.12.4.3 execute()

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

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

Returns

The current Program object.

5.12.4.4 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.

5.12.4.5 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.

5.12.4.6 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.cpp

5.13 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.

5.13.1 Member Function Documentation

5.13.1.1 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.

5.13.1.2 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.

5.13.1.3 recycle()

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

Recycle a memory location for an object T.

Parameters

obj	The memory location to recycle.

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

- include/singletonObjectPool.hpp

5.14 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.

5.14.1 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).

5.14.2 Constructor & Destructor Documentation

5.14.2.1 TangBase()

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

The constructor.

Isn't it glorious.

5.14.3 Member Function Documentation

5.14.3.1 compileScript()

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

Compile the provided source code as a script and return a Program.

Parameters

script	The Tang script to be compiled.

Returns

The Program object representing the compiled script.

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

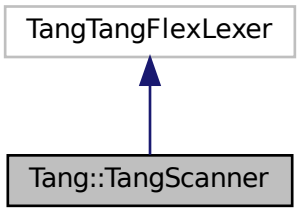
- include/tangBase.hpp
- src/tangBase.cpp

5.15 Tang::TangScanner Class Reference

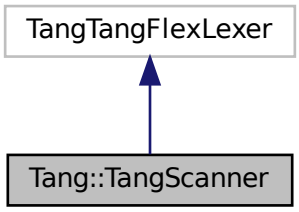
The Flex lexer class for the main Tang language.

`#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.

5.15.1 Detailed Description

The Flex lexer class for the main Tang language.

Flex requires that our lexer class inherit from yyFlexLexer, an "intermediate" class whose real name is "TangTang↩ FlexLexer". We are subclassing it so that we can override the return type of get_next_token(), for compatibility with Bison 3 tokens.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 TangScanner()

```
Tang::TangScanner::TangScanner (
            std::istream & arg_yyin,
            std::ostream & arg_yyout )  [inline]
```

The constructor for the Scanner.

The design of the Flex lexer is to tokenize the contents of an input stream, and to write any error messages to an output stream. In our implementation, however, errors are returned differently, so the output stream is never used. It's presence is retained, however, in case it is needed in the future.

For now, the general approach should be to supply the input as a string stream, and to use std::cout as the output.

Parameters

arg_yyin	The input stream to be tokenized
arg_yyout	The output stream (not currently used)

5.15.3 Member Function Documentation

5.15.3.1 get_next_token()

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

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

Returns

A Bison 3 token representing the lexeme that was recognized.

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

- include/tangScanner.hpp

Chapter 6

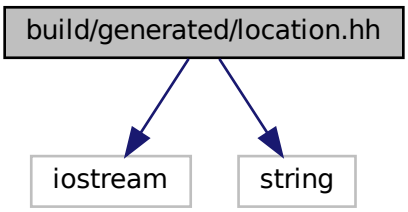
File Documentation

6.1 build/generated/location.hh File Reference

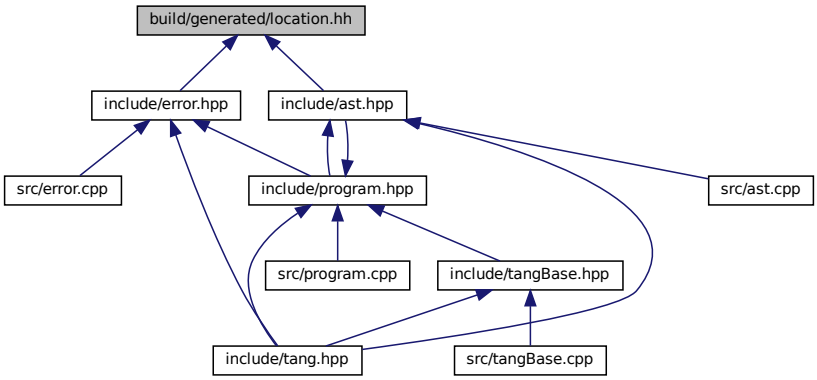
Define the Tang ::location class.

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

Include dependency graph for location.hh:



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



Classes

- class Tang::position

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

  Two points in a source file.

Macros

- #define **YY_NULLPTR** ((void∗)0)

Functions

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

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

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

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

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

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

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

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

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

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

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

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

  Intercept output stream redirection.

6.1.1 Detailed Description

Define the Tang ::location class.

6.1.2 Function Documentation

6.1.2.1 operator<<() [1/2]

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

Intercept output stream redirection.

Parameters

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

Avoid duplicate information.

6.1.2.2 operator<<() [2/2]

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

Intercept output stream redirection.

Parameters

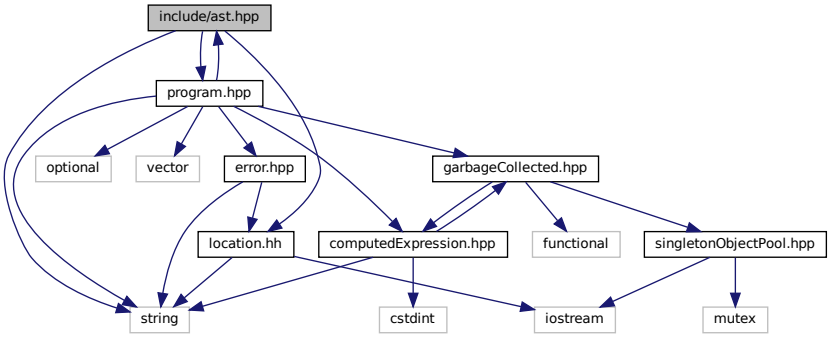
ostr	the destination output stream
pos	a reference to the position to redirect

6.2 include/ast.hpp File Reference

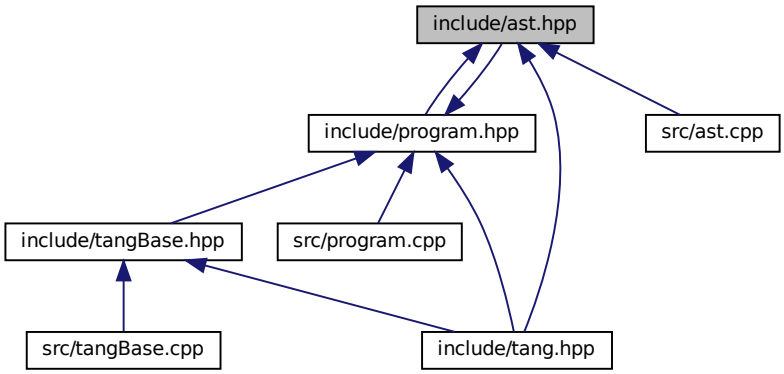
Define the Tang::AstNode and its associated/derivative classes.

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

Include dependency graph for ast.hpp:



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



Classes

- class Tang::AstNode

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

  An AstNode that represents an integer literal.
- class Tang::AstNodeFloat

  An AstNode that represents an float literal.
- class Tang::AstNodeAdd

  An AstNode that represents a "+" expression.

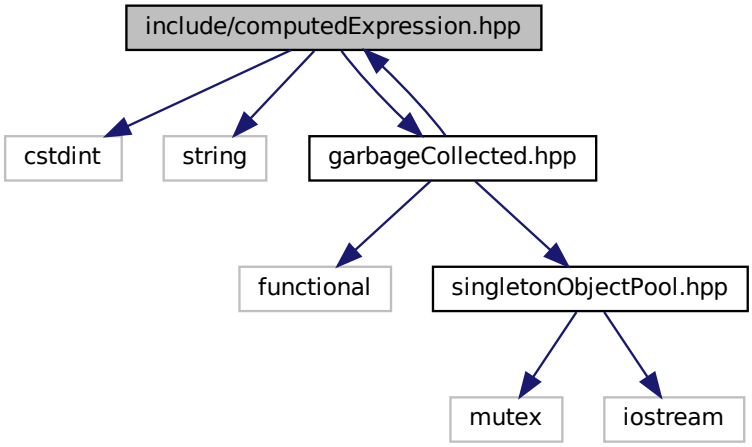
6.2.1 Detailed Description

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

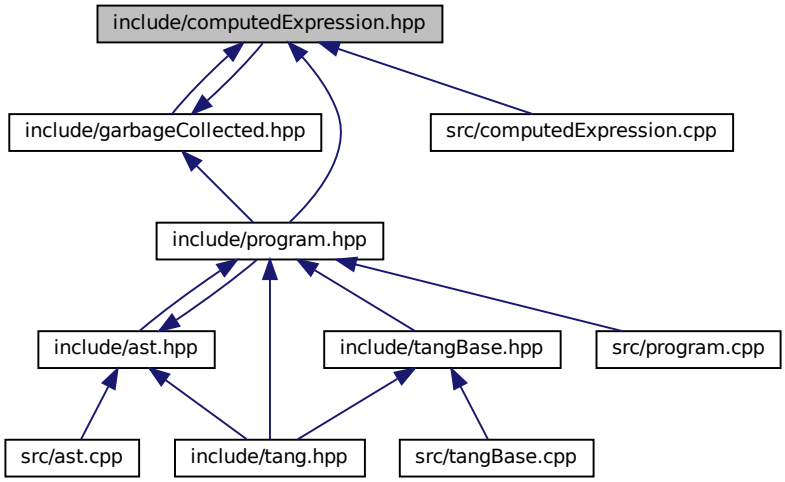
6.3 include/computedExpression.hpp File Reference

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

Include dependency graph for computedExpression.hpp:



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



Classes

- class Tang::ComputedExpression

Represents the result of a computation that has been executed.

- class Tang::ComputedExpressionInteger

  Represents an Integer that is the result of a computation.

- class Tang::ComputedExpressionFloat

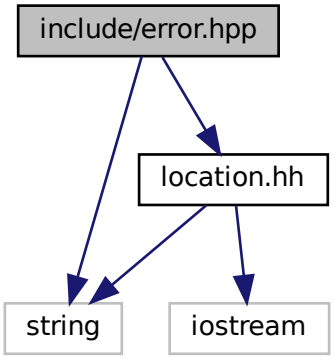
  Represents a Float that is the result of a computation.

6.4 include/error.hpp File Reference

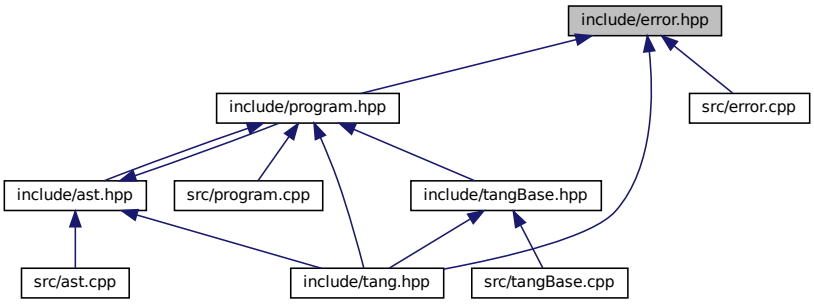
Define the Tang::Error class used to describe syntax and runtime errors.

```
#include <string>
#include "location.hh"
```

Include dependency graph for error.hpp:



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



Classes

- class Tang::Error

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

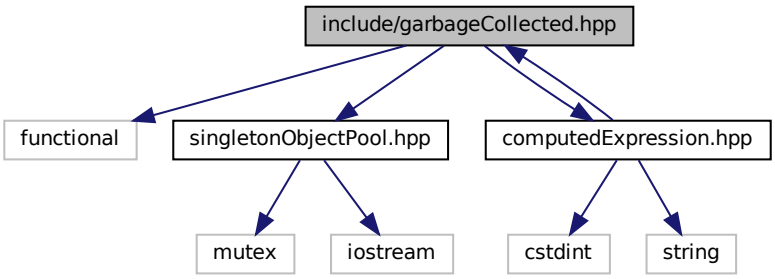
6.4.1 Detailed Description

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

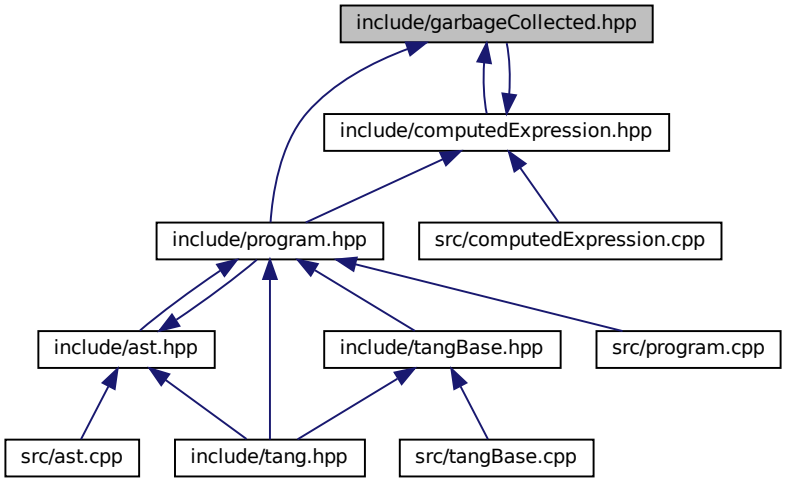
6.5 include/garbageCollected.hpp File Reference

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

Include dependency graph for garbageCollected.hpp:



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



Classes

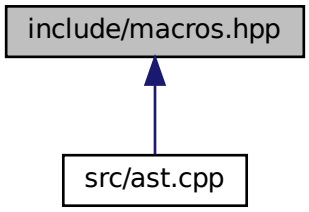
- class Tang::GarbageCollected

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

6.6 include/macros.hpp File Reference

Contains generic macros.

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



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.

6.6.1 Detailed Description

Contains generic macros.

6.6.2 Macro Definition Documentation

6.6.2.1 TANG_UNUSED

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

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

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

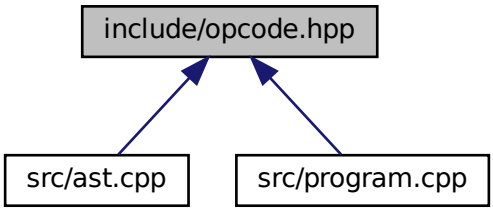
Parameters

x	The argument to be ignored.

6.7 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 }

6.7.1 Detailed Description

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

6.7.2 Enumeration Type Documentation

6.7.2.1 Opcode

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

Enumerator

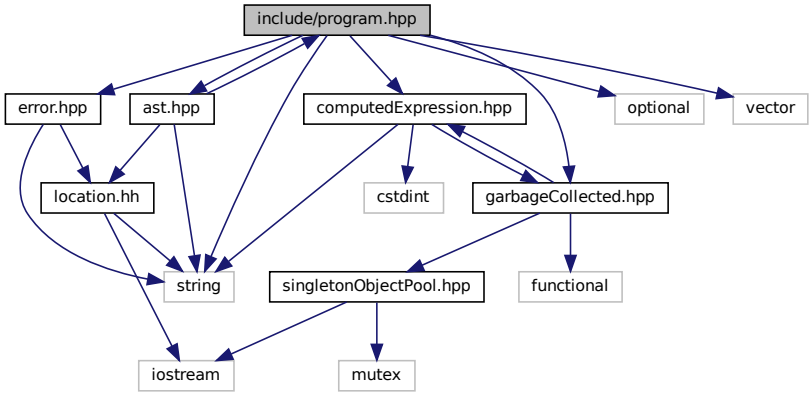
INTEGER	Push an integer onto the stack.
FLOAT	Push a floating point number onto the stack.
ADD	Pop rhs, pop lhs, push lhs + rhs.

6.8 include/program.hpp File Reference

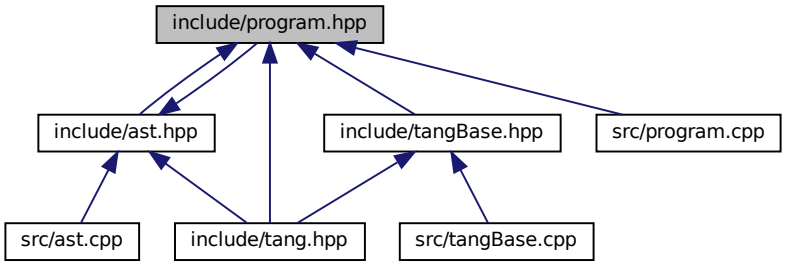
Define the Tang::Program class used to compile and execute source code.

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

Include dependency graph for program.hpp:



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



Classes

- class Tang::Program

  Represents a compiled script or template that may be executed.

Typedefs

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

  Contains the Opcodes of a compiled program.

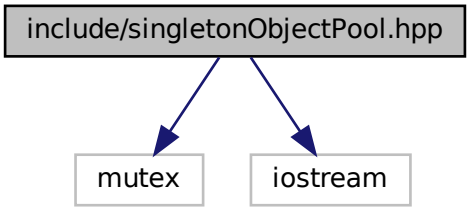
6.8.1 Detailed Description

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

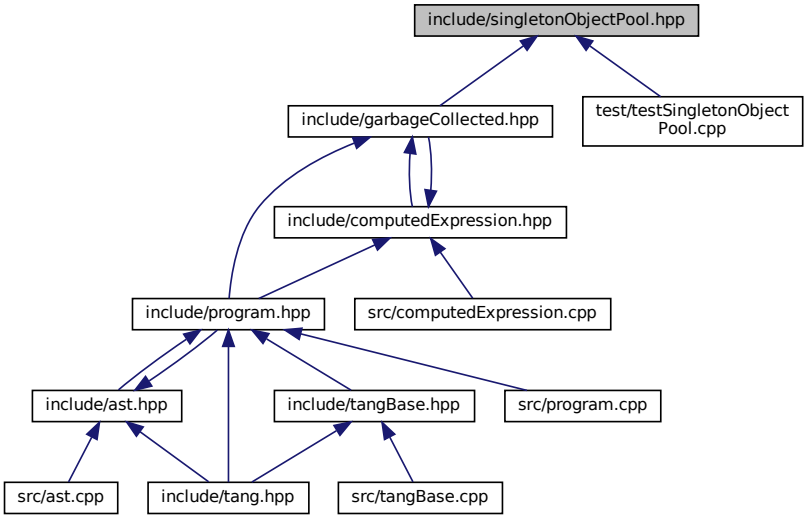
6.9 include/singletonObjectPool.hpp File Reference

```
#include <mutex>
#include <iostream>
```

Include dependency graph for singletonObjectPool.hpp:



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



Classes

- class Tang::SingletonObjectPool< T >

Macros

- #define GROW 1024

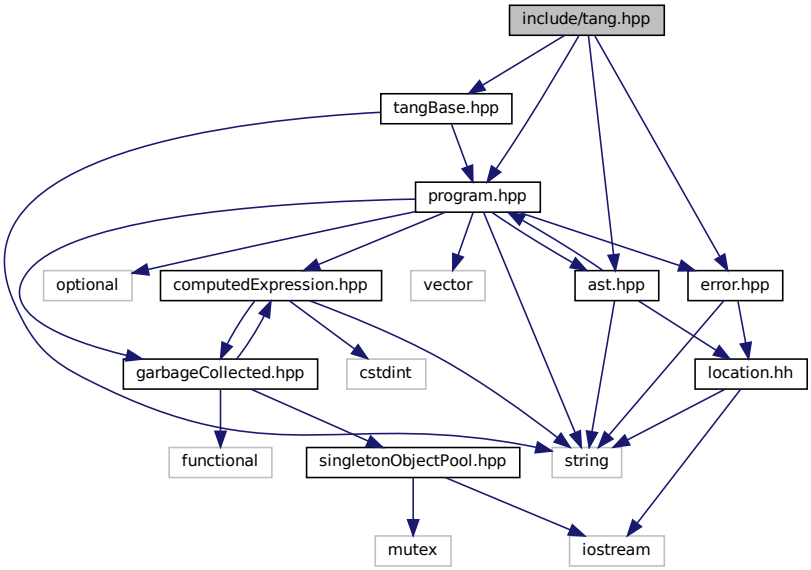
  The threshold size to use when allocating blocks of data, measured in the number of instances of the object type.

6.10 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 "ast.hpp"
#include "error.hpp"
#include "program.hpp"
```

Include dependency graph for tang.hpp:



6.10.1 Detailed Description

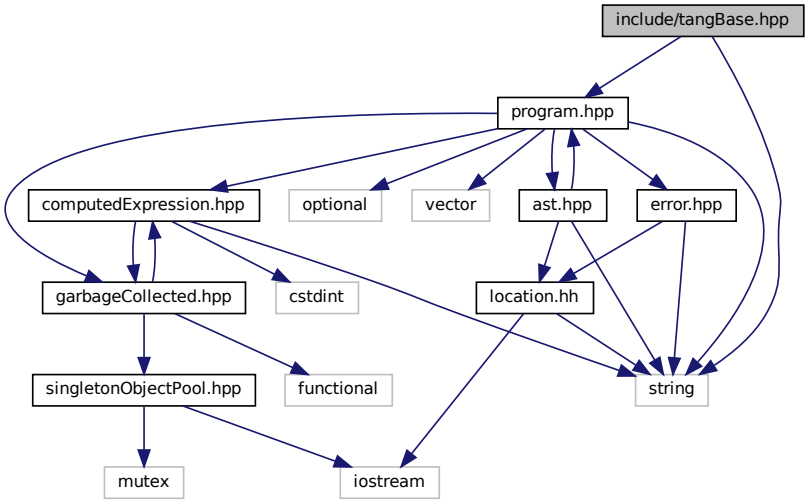
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

6.11 include/tangBase.hpp File Reference

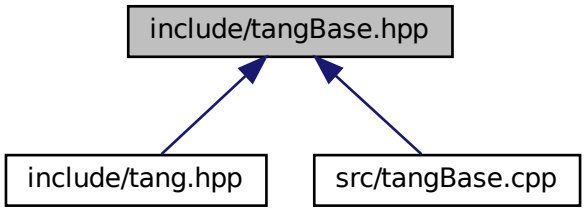
Defines the Tang::TangBase class used to interact with Tang.

```
#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.

6.11.1 Detailed Description

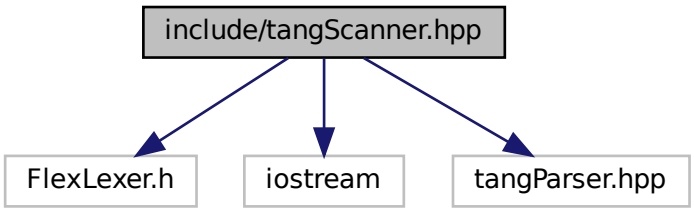
Defines the Tang::TangBase class used to interact with Tang.

6.12 include/tangScanner.hpp File Reference

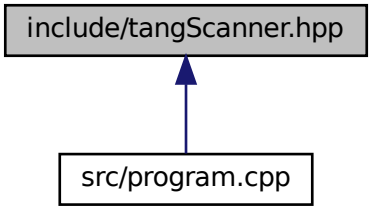
Defines the Tang::TangScanner used to tokenize a Tang script.

```
#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()

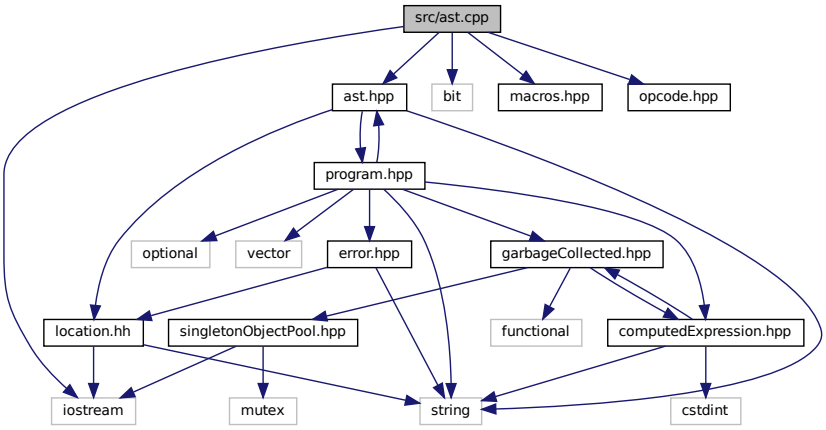
6.12.1 Detailed Description

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

6.13 src/ast.cpp File Reference

```
#include <iostream>
#include <bit>
#include "ast.hpp"
#include "macros.hpp"
#include "opcode.hpp"
```

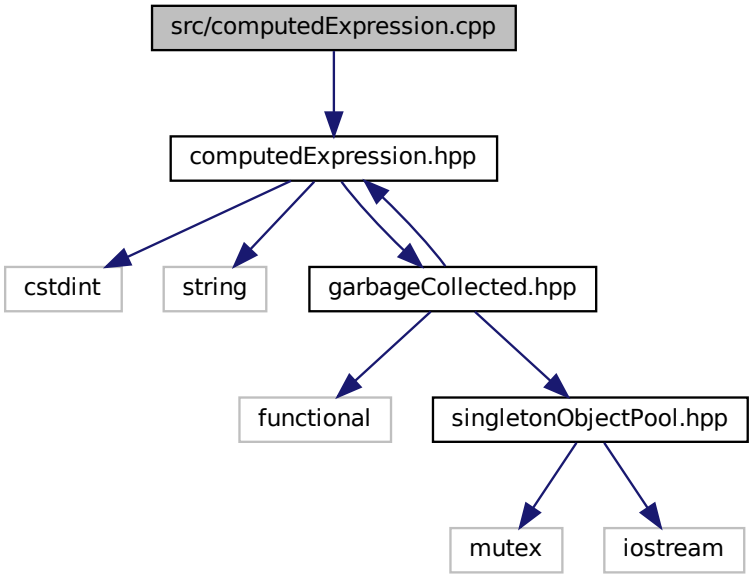
Include dependency graph for ast.cpp:



6.14 src/computedExpression.cpp File Reference

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

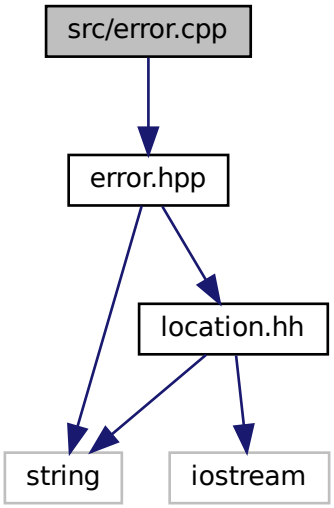
Include dependency graph for computedExpression.cpp:



6.15 src/error.cpp File Reference

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

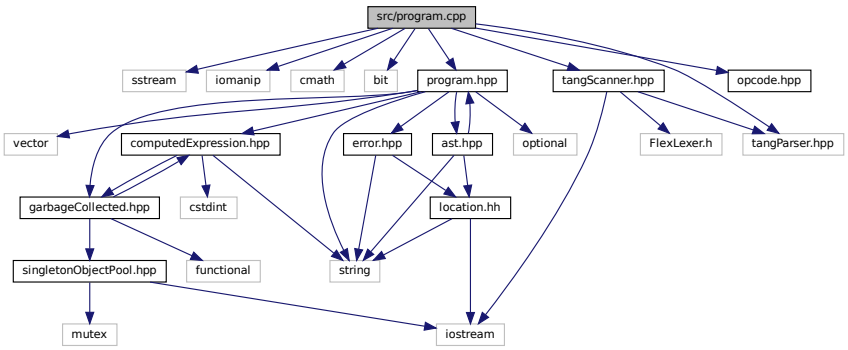
Include dependency graph for error.cpp:



6.16 src/program.cpp File Reference

```
#include <sstream>
#include <iomanip>
#include <cmath>
#include <bit>
#include "program.hpp"
#include "tangScanner.hpp"
#include "tangParser.hpp"
#include "opcode.hpp"
```

Include dependency graph for program.cpp:



Macros

- #define DUMPPROGRAMCHECK(x)

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

6.16.1 Macro Definition Documentation

6.16.1.1 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.

6.16.1.2 EXECUTEPROGRAMCHECK

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

Value:

```
  if (this->bytecode.size() < (pc + (x))) { \
    /* TODO push an error on to the stack! */ \
    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.

6.16.1.3 STACKCHECK

```
#define STACKCHECK(
              x )
```

Value:

```
if (stack.size() < (fp + (x))) { \
  /* TODO push an error on to the stack! */ \
  pc = this->bytecode.size(); \
  break; \
}
```

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

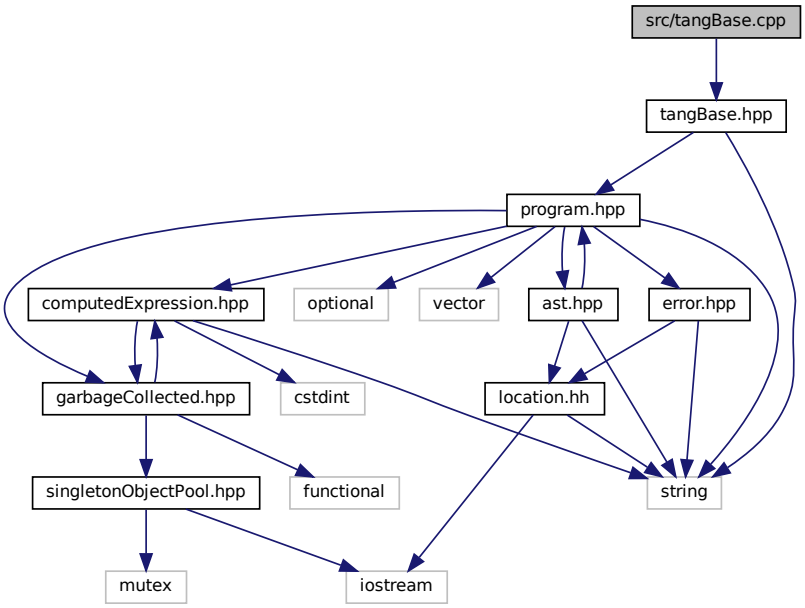
Parameters

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

6.17 src/tangBase.cpp File Reference

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

Include dependency graph for tangBase.cpp:

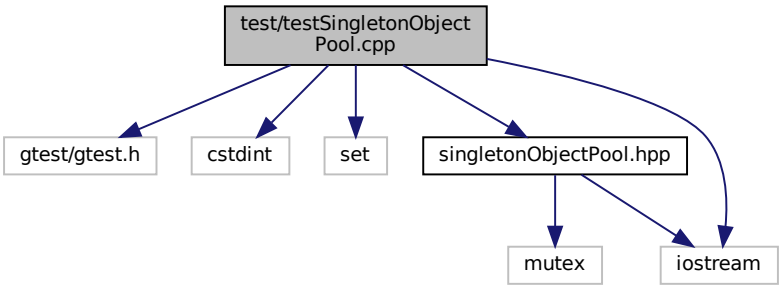


6.18 test/testSingletonObjectPool.cpp File Reference

```
#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, RecycledObjectIsReused)
- **TEST** (Get, SuccessiveCallsAreSequential)
- **TEST** (Get, KeepsGeneratingDifferentPointers)
- **TEST** (Recycle, WorksAfterLargeNumberOfAllocations)
- int **main** (int argc, char ∗∗argv)

Index

__add
    Tang::ComputedExpression, 21
    Tang::ComputedExpressionFloat, 25
    Tang::ComputedExpressionInteger, 28
~GarbageCollected
    Tang::GarbageCollected, 34

ADD
    opcode.hpp, 59
addBytecode
    Tang::Program, 44
AstNode
    Tang::AstNode, 11
AstNodeAdd
    Tang::AstNodeAdd, 14
AstNodeFloat
    Tang::AstNodeFloat, 17
AstNodeInteger
    Tang::AstNodeInteger, 20

build/generated/location.hh, 51

CodeType
    Tang::Program, 43
compileScript
    Tang::TangBase, 48
ComputedExpressionFloat
    Tang::ComputedExpressionFloat, 24
ComputedExpressionInteger
    Tang::ComputedExpressionInteger, 28

dump
    Tang::ComputedExpression, 22
    Tang::ComputedExpressionFloat, 25
    Tang::ComputedExpressionInteger, 29
dumpBytecode
    Tang::Program, 44
DUMPPROGRAMCHECK
    program.cpp, 68

Error
    Tang::Error, 32
execute
    Tang::Program, 45
EXECUTEPROGRAMCHECK
    program.cpp, 68

FLOAT
    opcode.hpp, 59

GarbageCollected
    Tang::GarbageCollected, 34
get
    Tang::SingletonObjectPool< T >, 46
get_next_token
    Tang::TangScanner, 50
getAst
    Tang::Program, 45
getCode
    Tang::Program, 45
getInstance
    Tang::SingletonObjectPool< T >, 46
getResult
    Tang::Program, 45

include/ast.hpp, 53
include/computedExpression.hpp, 55
include/error.hpp, 56
include/garbageCollected.hpp, 57
include/macros.hpp, 58
include/opcode.hpp, 59
include/program.hpp, 59
include/singletonObjectPool.hpp, 61
include/tang.hpp, 62
include/tangBase.hpp, 62
include/tangScanner.hpp, 64
INTEGER
    opcode.hpp, 59
is_equal
    Tang::ComputedExpression, 22
    Tang::ComputedExpressionFloat, 25, 26
    Tang::ComputedExpressionInteger, 29

location.hh
    operator<<, 52, 53

macros.hpp
    TANG_UNUSED, 58
make
    Tang::GarbageCollected, 36
makeCopy
    Tang::AstNode, 11
    Tang::AstNodeAdd, 14
    Tang::AstNodeFloat, 17
    Tang::AstNodeInteger, 20
    Tang::ComputedExpression, 23
    Tang::ComputedExpressionFloat, 26
    Tang::ComputedExpressionInteger, 30

Opcode
    opcode.hpp, 59

opcode.hpp
ADD, 59
FLOAT, 59
INTEGER, 59
Opcode, 59
operator<<
location.hh, 52, 53
Tang::GarbageCollected, 38
operator*
Tang::GarbageCollected, 36
operator->
Tang::GarbageCollected, 36
operator=
Tang::GarbageCollected, 37
operator==
Tang::GarbageCollected, 38

Program
Tang::Program, 44
program.cpp
DUMPPROGRAMCHECK, 68
EXECUTEPROGRAMCHECK, 68
STACKCHECK, 69

recycle
Tang::SingletonObjectPool< T >, 47

Script
Tang::Program, 44
src/ast.cpp, 65
src/computedExpression.cpp, 66
src/error.cpp, 67
src/program.cpp, 67
src/tangBase.cpp, 69
STACKCHECK
program.cpp, 69

Tang::AstNode, 9
AstNode, 11
makeCopy, 11
Tang::AstNodeAdd, 12
AstNodeAdd, 14
makeCopy, 14
Tang::AstNodeFloat, 15
AstNodeFloat, 17
makeCopy, 17
Tang::AstNodeInteger, 18
AstNodeInteger, 20
makeCopy, 20
Tang::ComputedExpression, 21
__add, 21
dump, 22
is_equal, 22
makeCopy, 23
Tang::ComputedExpressionFloat, 23
__add, 25
ComputedExpressionFloat, 24
dump, 25
is_equal, 25, 26
makeCopy, 26
Tang::ComputedExpressionInteger, 27
__add, 28
ComputedExpressionInteger, 28
dump, 29
is_equal, 29
makeCopy, 30
Tang::Error, 31
Error, 32
Tang::GarbageCollected, 32
~GarbageCollected, 34
GarbageCollected, 34
make, 36
operator<<, 38
operator*, 36
operator->, 36
operator=, 37
operator==, 38
Tang::location, 39
Tang::position, 41
Tang::Program, 42
addBytecode, 44
CodeType, 43
dumpBytecode, 44
execute, 45
getAst, 45
getCode, 45
getResult, 45
Program, 44
Script, 44
Template, 44
Tang::SingletonObjectPool< T >, 46
get, 46
getInstance, 46
recycle, 47
Tang::TangBase, 47
compileScript, 48
TangBase, 48
Tang::TangScanner, 49
get_next_token, 50
TangScanner, 50
TANG_UNUSED
macros.hpp, 58
TangBase
Tang::TangBase, 48
TangScanner
Tang::TangScanner, 50
Template
Tang::Program, 44
test/testSingletonObjectPool.cpp, 69