Tang

0.1

Generated by Doxygen 1.9.1

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

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

Chapter 5

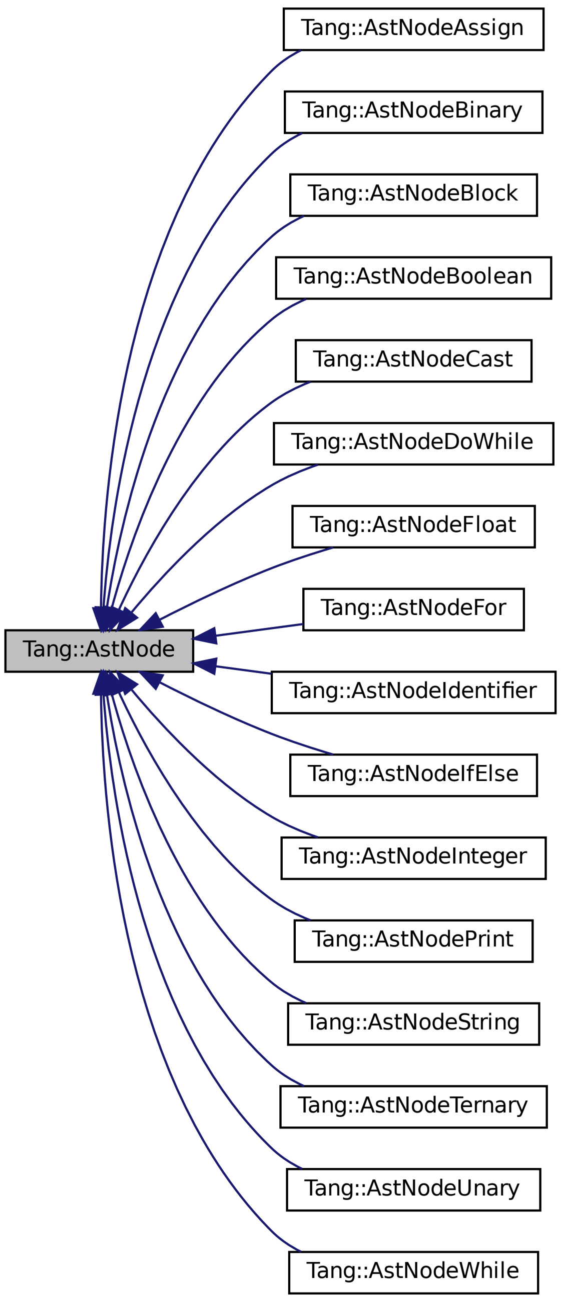
Class Documentation

5.1 Tang::AstNode Class Reference

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

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

Inheritance diagram for Tang::AstNode:



Public Member Functions

- AstNode (Tang::location location)

  The generic constructor.
- virtual ∼AstNode ()

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

Return a string that describes the contents of the node.

- virtual void compile (Tang::Program &program) const

  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const

  Compile a list of all string constants in the scope.

5.1.1 Detailed Description

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

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

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AstNode()

```
AstNode::AstNode (
            Tang::location location )
```

The generic constructor.

It should never be called on its own.

Parameters

location	The location associated with this node.

5.1.3 Member Function Documentation

5.1.3.1 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.1.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.1.3.3 compile()

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

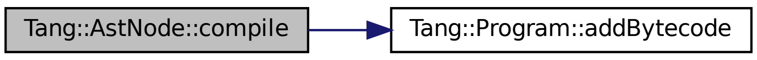
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeInteger, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeFloat, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBoolean, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

Here is the call graph for this function:



5.1.3.4 dump()

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

Return a string that describes the contents of the node.

Parameters

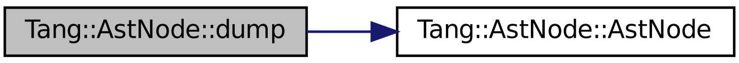
indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeInteger, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeFloat, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBoolean, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

Here is the call graph for this function:



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

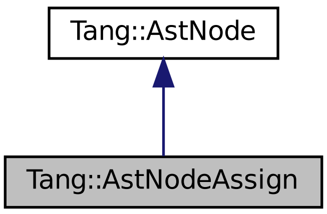
- include/astNode.hpp
- src/astNode.cpp

5.2 Tang::AstNodeAssign Class Reference

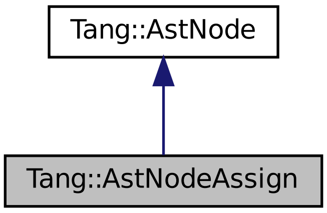
An AstNode that represents a binary expression.

`#include <astNodeAssign.hpp>`

Inheritance diagram for Tang::AstNodeAssign:



Collaboration diagram for Tang::AstNodeAssign:



Public Member Functions

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

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

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

    Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const override

    Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

    Compile a list of all string constants in the scope.

5.2.1 Detailed Description

An AstNode that represents a binary expression.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AstNodeAssign()

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

The constructor.

Parameters

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

5.2.3 Member Function Documentation

5.2.3.1 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.2.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.2.3.3 compile()

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

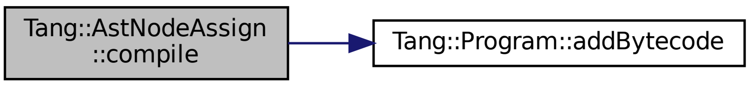
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.2.3.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

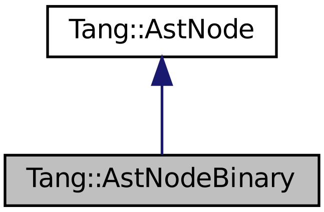
- include/astNodeAssign.hpp
- src/astNodeAssign.cpp

5.3 Tang::AstNodeBinary Class Reference

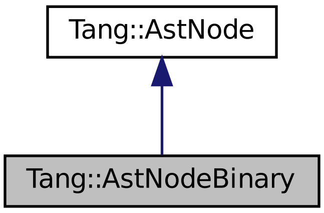
An AstNode that represents a binary expression.

```
#include <astNodeBinary.hpp>
```

Inheritance diagram for Tang::AstNodeBinary:



Collaboration diagram for Tang::AstNodeBinary:



Public Types

- enum Operation {
  Add , Subtract , Multiply , Divide ,
  Modulo , LessThan , LessThanEqual , GreaterThan ,
  GreaterThanEqual , Equal , NotEqual , And ,
  Or }

  Indicates the type of binary expression that this node represents.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

5.3.1 Detailed Description

An AstNode that represents a binary expression.

5.3.2 Member Enumeration Documentation

5.3.2.1 Operation

```
enum Tang::AstNodeBinary::Operation
```

Indicates the type of binary expression that this node represents.

Enumerator

Add	Indicates lhs + rhs.
Subtract	Indicates lhs - rhs.
Multiply	Indicates lhs ∗ rhs.
Divide	Indicates lhs / rhs.
Modulo	Indicates lhs % rhs.
LessThan	Indicates lhs < rhs.
LessThanEqual	Indicates lhs <= rhs.
GreaterThan	Indicates lhs > rhs.
GreaterThanEqual	Indicates lhs >= rhs.
Equal	Indicates lhs == rhs.
NotEqual	Indicates lhs != rhs.
And	Indicates lhs && rhs with short-circuit evaluation.
Or	Indicates lhs || rhs with short-circuit evaluation.

5.3.3 Constructor & Destructor Documentation

5.3.3.1 AstNodeBinary()

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

The constructor.

Parameters

op	The Tang::AstNodeBinary::Operation to perform.
lhs	The left hand side expression.
rhs	The right hand side expression.
location	The location associated with the expression.

5.3.4 Member Function Documentation

5.3.4.1 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.3.4.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.3.4.3 compile()

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

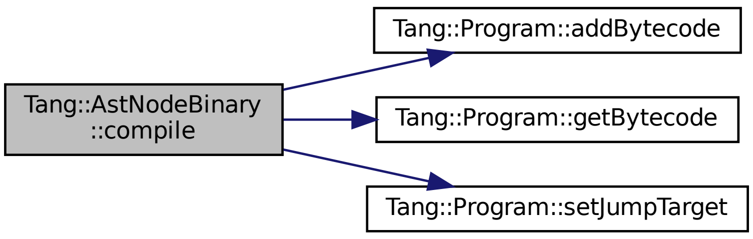
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeBinary ::compile
Tang::Program::addBytecode
Tang::Program::getBytecode
Tang::Program::setJumpTarget

5.3.4.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

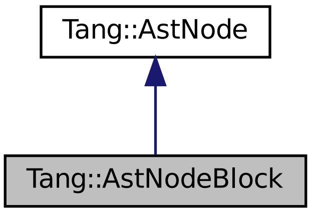
- include/astNodeBinary.hpp
- src/astNodeBinary.cpp

5.4 Tang::AstNodeBlock Class Reference

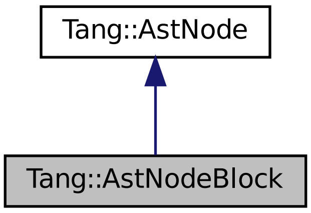
An AstNode that represents a code block.

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

Inheritance diagram for Tang::AstNodeBlock:



Collaboration diagram for Tang::AstNodeBlock:



Public Member Functions

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

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

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

    Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const override

    Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

    Compile a list of all string constants in the scope.

5.4.1 Detailed Description

An AstNode that represents a code block.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 AstNodeBlock()

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

The constructor.

Parameters

statements	The statements of the code block.
location	The location associated with the expression.

5.4.3 Member Function Documentation

5.4.3.1 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.4.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.4.3.3 compile()

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

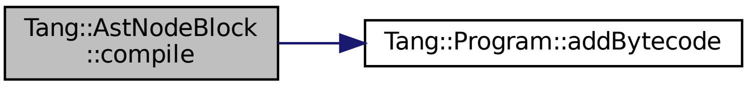
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.4.3.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

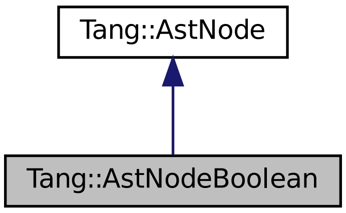
- include/astNodeBlock.hpp
- src/astNodeBlock.cpp

5.5 Tang::AstNodeBoolean Class Reference

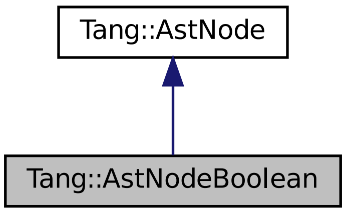
An AstNode that represents a boolean literal.

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

Inheritance diagram for Tang::AstNodeBoolean:



Collaboration diagram for Tang::AstNodeBoolean:



Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const

  Compile a list of all string constants in the scope.

5.5.1 Detailed Description

An AstNode that represents a boolean literal.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 AstNodeBoolean()

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

The constructor.

Parameters

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

5.5.3 Member Function Documentation

5.5.3.1 collectIdentifiers()

```
void AstNode::collectIdentifiers (
             Program & program ) const  [virtual], [inherited]
```

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.5.3.2 collectStrings()

```
void AstNode::collectStrings (
             Program & program ) const  [virtual], [inherited]
```

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.5.3.3 compile()

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

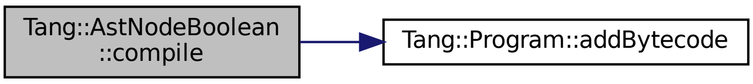
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.5.3.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

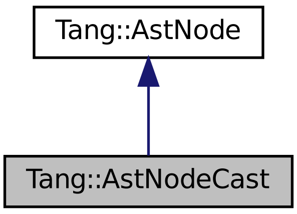
- include/astNodeBoolean.hpp
- src/astNodeBoolean.cpp

5.6 Tang::AstNodeCast Class Reference

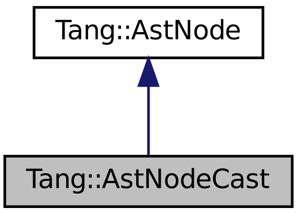
An AstNode that represents a typecast of an expression.

```
#include <astNodeCast.hpp>
```

Inheritance diagram for Tang::AstNodeCast:



Collaboration diagram for Tang::AstNodeCast:



Public Types

- enum Type { Integer , Float , Boolean }

  The possible types that can be cast to.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const

  Compile a list of all string constants in the scope.

5.6.1 Detailed Description

An AstNode that represents a typecast of an expression.

5.6.2 Member Enumeration Documentation

5.6.2.1 Type

```
enum Tang::AstNodeCast::Type
```

The possible types that can be cast to.

Enumerator

Integer	Cast to a Tang::ComputedExpressionInteger.
Float	Cast to a Tang::ComputedExpressionFloat.
Boolean	Cast to a Tang::ComputedExpressionBoolean.

5.6.3 Constructor & Destructor Documentation

5.6.3.1 AstNodeCast()

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

The constructor.

Parameters

targetType	The target type that the expression will be cast to.
expression	The expression to be typecast.
location	The location associated with this node.

5.6.4 Member Function Documentation

5.6.4.1 collectIdentifiers()

```
void AstNode::collectIdentifiers (
            Program & program ) const  [virtual], [inherited]
```

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.6.4.2 collectStrings()

```
void AstNode::collectStrings (
            Program & program ) const  [virtual], [inherited]
```

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.6.4.3 compile()

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

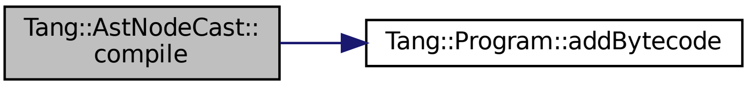
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.6.4.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

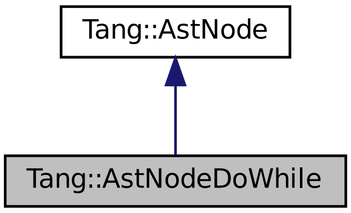
- include/astNodeCast.hpp
- src/astNodeCast.cpp

5.7 Tang::AstNodeDoWhile Class Reference

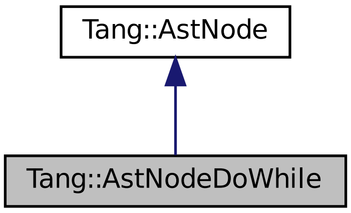
An AstNode that represents a do..while statement.

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

Inheritance diagram for Tang::AstNodeDoWhile:



Collaboration diagram for Tang::AstNodeDoWhile:



Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

5.7.1 Detailed Description

An AstNode that represents a do..while statement.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 AstNodeDoWhile()

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

The constructor.

Parameters

condition	The expression which determines whether the thenBlock or elseBlock is executed.
codeBlock	The statement executed when the condition is true.
location	The location associated with the expression.

5.7.3 Member Function Documentation

5.7.3.1 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.7.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.7.3.3 compile()

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

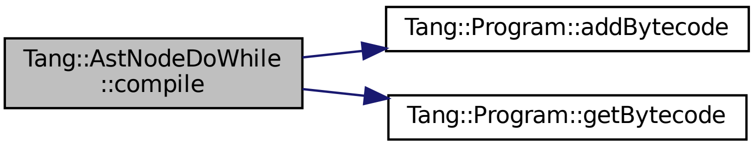
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeDoWhile ::compile
Tang::Program::addBytecode
Tang::Program::getBytecode

5.7.3.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

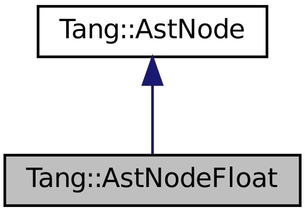
- include/astNodeDoWhile.hpp
- src/astNodeDoWhile.cpp

5.8 Tang::AstNodeFloat Class Reference

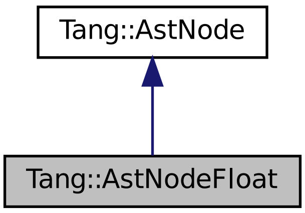
An AstNode that represents an float literal.

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

Inheritance diagram for Tang::AstNodeFloat:



Collaboration diagram for Tang::AstNodeFloat:



Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const

  Compile a list of all string constants in the scope.

5.8.1 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.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 AstNodeFloat()

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

The constructor.

Parameters

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

5.8.3 Member Function Documentation

5.8.3.1 collectIdentifiers()

```
void AstNode::collectIdentifiers (
            Program & program ) const  [virtual], [inherited]
```

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.8.3.2 collectStrings()

```
void AstNode::collectStrings (
            Program & program ) const  [virtual], [inherited]
```

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.8.3.3 compile()

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

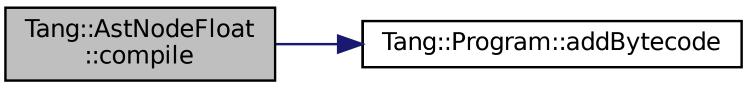
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.8.3.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

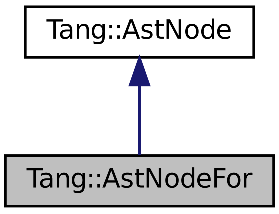
- include/astNodeFloat.hpp
- src/astNodeFloat.cpp

5.9 Tang::AstNodeFor Class Reference

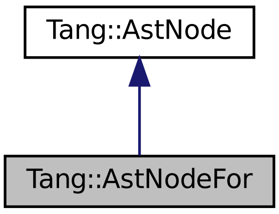
An AstNode that represents an if() statement.

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

Inheritance diagram for Tang::AstNodeFor:



Collaboration diagram for Tang::AstNodeFor:



Public Member Functions

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

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

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

    Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const override

    Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

    Compile a list of all string constants in the scope.

5.9.1 Detailed Description

An AstNode that represents an if() statement.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 AstNodeFor()

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

The constructor.

Parameters

initialization	The expression to be executed first.
condition	The expression which determines whether the codeBlock is executed.
increment	The expression to be executed after each codeBlock.
codeBlock	The statement executed when the condition is true.
location	The location associated with the expression.

5.9.3 Member Function Documentation

5.9.3.1 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.9.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.9.3.3 compile()

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

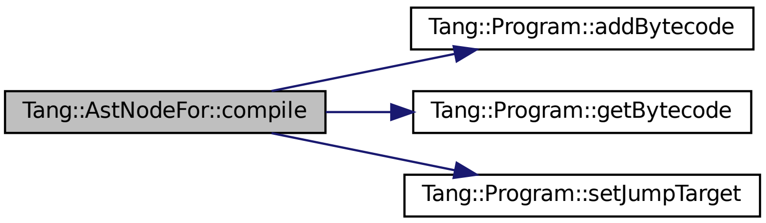
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeFor::compile
Tang::Program::addBytecode
Tang::Program::getBytecode
Tang::Program::setJumpTarget

5.9.3.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

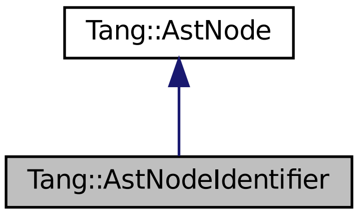
- include/astNodeFor.hpp
- src/astNodeFor.cpp

5.10 Tang::AstNodeIdentifier Class Reference

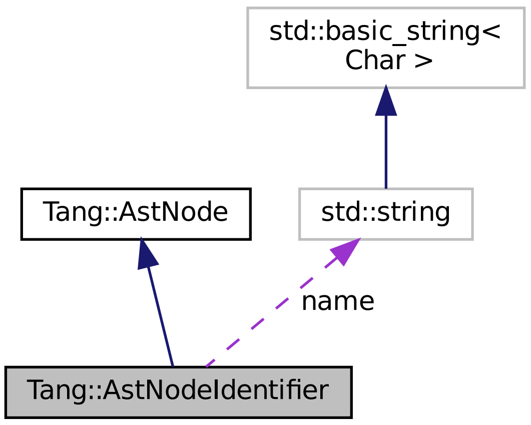
An AstNode that represents an identifier.

`#include <astNodeIdentifier.hpp>`

Inheritance diagram for Tang::AstNodeIdentifier:



Collaboration diagram for Tang::AstNodeIdentifier:



Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const

  Compile a list of all string constants in the scope.

Public Attributes

- std::string name
  The name of the identifier.

5.10.1 Detailed Description

An AstNode that represents an identifier.

Identifier names are represented by a string.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 AstNodeIdentifier()

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

The constructor.

Parameters

name	The name of the identifier
location	The location associated with the expression.

5.10.3 Member Function Documentation

5.10.3.1 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.10.3.2 collectStrings()

```
void AstNode::collectStrings (
            Program & program ) const  [virtual], [inherited]
```

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.10.3.3 compile()

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

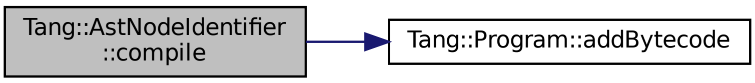
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.10.3.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

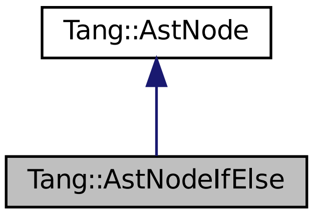
- include/astNodeIdentifier.hpp
- src/astNodeIdentifier.cpp

5.11 Tang::AstNodeIfElse Class Reference

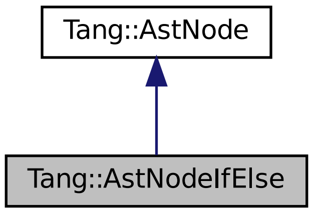
An AstNode that represents an if..else statement.

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

Inheritance diagram for Tang::AstNodeIfElse:



Collaboration diagram for Tang::AstNodeIfElse:



Public Member Functions

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

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

5.11.1 Detailed Description

An AstNode that represents an if..else statement.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 AstNodeIfElse() [1/2]

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

The constructor.

Parameters

condition	The expression which determines whether the thenBlock or elseBlock is executed.
thenBlock	The statement executed when the condition is true.
elseBlock	The statement executed when the condition is false.
location	The location associated with the expression.

5.11.2.2 AstNodeIfElse() [2/2]

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

The constructor.

Parameters

condition	The expression which determines whether the thenBlock or elseBlock is executed.
thenBlock	The statement executed when the condition is true.
location	The location associated with the expression.

5.11.3 Member Function Documentation

5.11.3.1 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.11.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.11.3.3 compile()

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

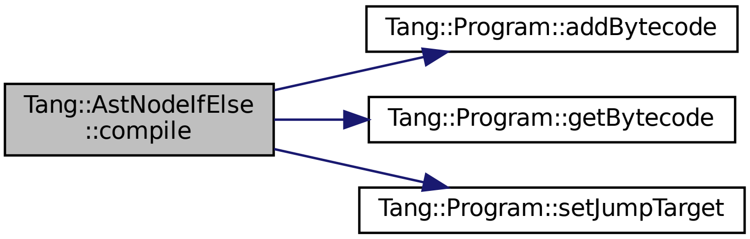
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeIfElse ::compile
Tang::Program::addBytecode
Tang::Program::getBytecode
Tang::Program::setJumpTarget

5.11.3.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

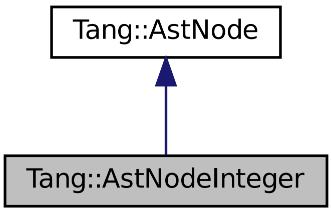
- include/astNodeIfElse.hpp
- src/astNodeIfElse.cpp

5.12 Tang::AstNodeInteger Class Reference

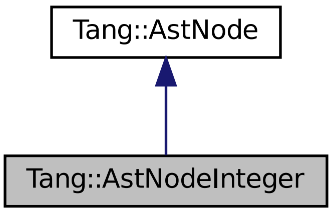
An AstNode that represents an integer literal.

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

Inheritance diagram for Tang::AstNodeInteger:



Collaboration diagram for Tang::AstNodeInteger:



Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const

  Compile a list of all string constants in the scope.

5.12.1 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.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 AstNodeInteger()

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

The constructor.

Parameters

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

5.12.3 Member Function Documentation

5.12.3.1 collectIdentifiers()

```
void AstNode::collectIdentifiers (
            Program & program ) const  [virtual], [inherited]
```

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.12.3.2 collectStrings()

```
void AstNode::collectStrings (
            Program & program ) const  [virtual], [inherited]
```

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.12.3.3 compile()

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

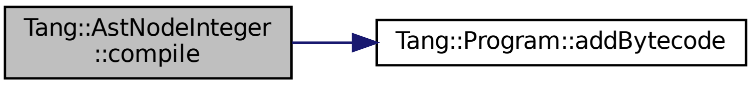
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.12.3.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

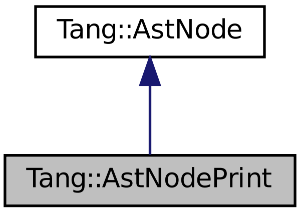
- include/astNodeInteger.hpp
- src/astNodeInteger.cpp

5.13 Tang::AstNodePrint Class Reference

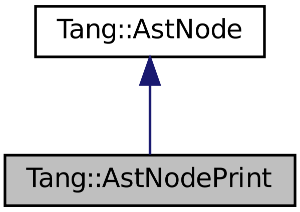
An AstNode that represents a print typeeration.

```
#include <astNodePrint.hpp>
```

Inheritance diagram for Tang::AstNodePrint:



Collaboration diagram for Tang::AstNodePrint:



Public Types

- enum Type { Default }

  The type of print() requested.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

5.13.1 Detailed Description

An AstNode that represents a print typeeration.

5.13.2 Member Enumeration Documentation

5.13.2.1 Type

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

The type of print() requested.

Enumerator

Default	Use the default print.

5.13.3 Constructor & Destructor Documentation

5.13.3.1 AstNodePrint()

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

The constructor.

Parameters

type	The Tang::AstNodePrint::Type being requested.
expression	The expression to be printed.
location	The location associated with the expression.

5.13.4 Member Function Documentation

5.13.4.1 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.13.4.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.13.4.3 compile()

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

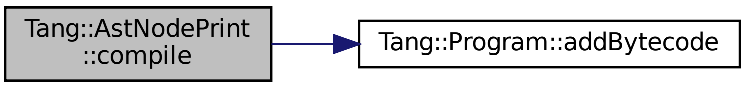
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.13.4.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

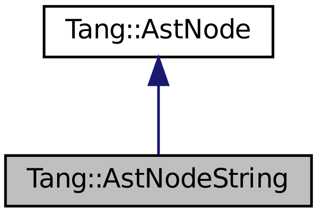
- include/astNodePrint.hpp
- src/astNodePrint.cpp

5.14 Tang::AstNodeString Class Reference

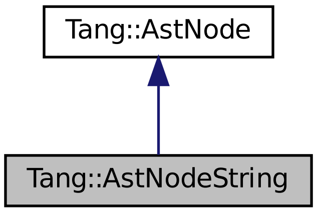
An AstNode that represents a string literal.

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

Inheritance diagram for Tang::AstNodeString:



Collaboration diagram for Tang::AstNodeString:



Public Member Functions

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

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

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

    Compile the ast of the provided Tang::Program.
- virtual void collectStrings (Program &program) const override

    Compile a list of all string constants in the scope.
- void compileLiteral (Tang::Program &program) const

    Compile the string and push it onto the stack.
- virtual void collectIdentifiers (Program &program) const

    Compile a list of all variables in the scope.

5.14.1 Detailed Description

An AstNode that represents a string literal.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 AstNodeString()

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

The constructor.

Parameters

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

5.14.3 Member Function Documentation

5.14.3.1 collectIdentifiers()

```
void AstNode::collectIdentifiers (
            Program & program ) const  [virtual], [inherited]
```

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.14.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.14.3.3 compile()

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

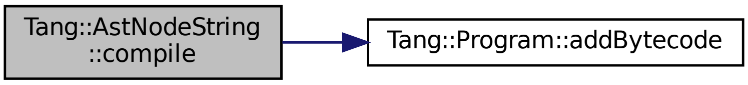
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.14.3.4 compileLiteral()

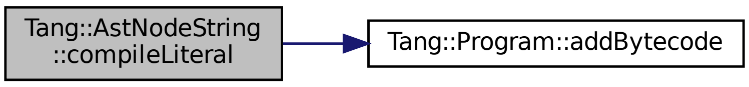
```
void AstNodeString::compileLiteral (
            Tang::Program & program ) const
```

Compile the string and push it onto the stack.

Parameters

program	The Program which will hold the generated Bytecode.

Here is the call graph for this function:


Tang::AstNodeString ::compileLiteral
Tang::Program::addBytecode

5.14.3.5 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

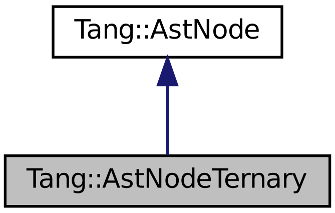
- include/astNodeString.hpp
- src/astNodeString.cpp

5.15 Tang::AstNodeTernary Class Reference

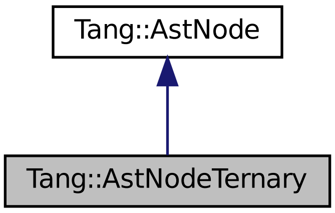
An AstNode that represents a ternary expression.

```
#include <astNodeTernary.hpp>
```

Inheritance diagram for Tang::AstNodeTernary:



Collaboration diagram for Tang::AstNodeTernary:



Public Member Functions

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

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

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

    Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const override

    Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

    Compile a list of all string constants in the scope.

5.15.1 Detailed Description

An AstNode that represents a ternary expression.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 AstNodeTernary()

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

The constructor.

Parameters

condition	The expression which determines whether the trueExpression or falseExpression is executed.
trueExpression	The expression executed when the condition is true.
falseExpression	The expression executed when the condition is false.
location	The location associated with the expression.

5.15.3 Member Function Documentation

5.15.3.1 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.15.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.15.3.3 compile()

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

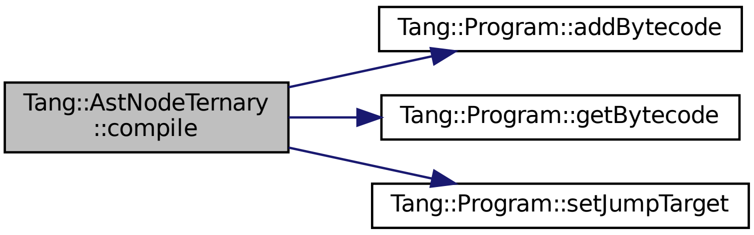
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeTernary ::compile
Tang::Program::addBytecode
Tang::Program::getBytecode
Tang::Program::setJumpTarget

5.15.3.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

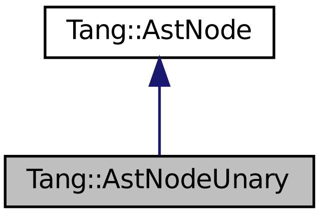
- include/astNodeTernary.hpp
- src/astNodeTernary.cpp

5.16 Tang::AstNodeUnary Class Reference

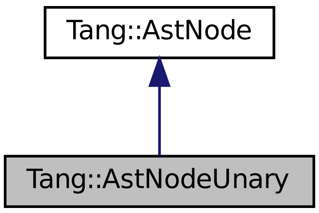
An AstNode that represents a unary negation.

```
#include <astNodeUnary.hpp>
```

Inheritance diagram for Tang::AstNodeUnary:



Collaboration diagram for Tang::AstNodeUnary:



Public Types

- enum Operator { Negative , Not }

    The type of operation.

Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

5.16.1 Detailed Description

An AstNode that represents a unary negation.

5.16.2 Member Enumeration Documentation

5.16.2.1 Operator

```
enum Tang::AstNodeUnary::Operator
```

The type of operation.

Enumerator

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

5.16.3 Constructor & Destructor Documentation

5.16.3.1 AstNodeUnary()

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

The constructor.

Parameters

op	The Tang::AstNodeUnary::Operator to apply to the operand.
operand	The expression to be operated on.
location	The location associated with the expression.

5.16.4 Member Function Documentation

5.16.4.1 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.16.4.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.16.4.3 compile()

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

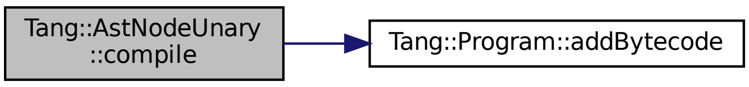
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.16.4.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

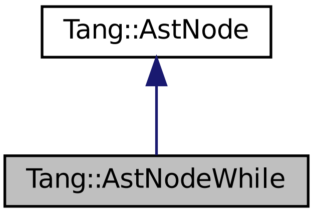
- include/astNodeUnary.hpp
- src/astNodeUnary.cpp

5.17 Tang::AstNodeWhile Class Reference

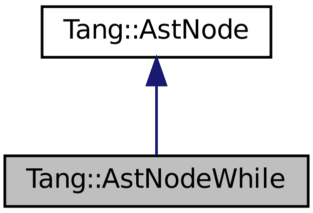
An AstNode that represents a while statement.

```
#include <astNodeWhile.hpp>
```

Inheritance diagram for Tang::AstNodeWhile:



Collaboration diagram for Tang::AstNodeWhile:



Public Member Functions

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

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

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

  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

5.17.1 Detailed Description

An AstNode that represents a while statement.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 AstNodeWhile()

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

The constructor.

Parameters

condition	The expression which determines whether the thenBlock or elseBlock is executed.
codeBlock	The statement executed when the condition is true.
location	The location associated with the expression.

5.17.3 Member Function Documentation

5.17.3.1 collectIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.17.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.17.3.3 compile()

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

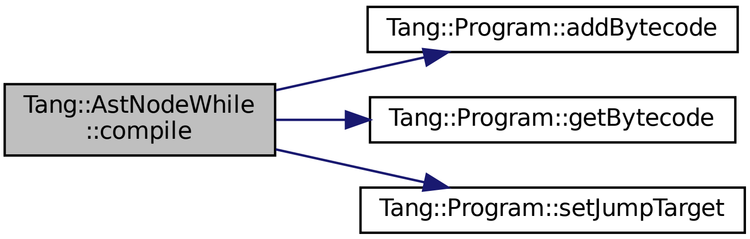
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeWhile ::compile
Tang::Program::addBytecode
Tang::Program::getBytecode
Tang::Program::setJumpTarget

5.17.3.4 dump()

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

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

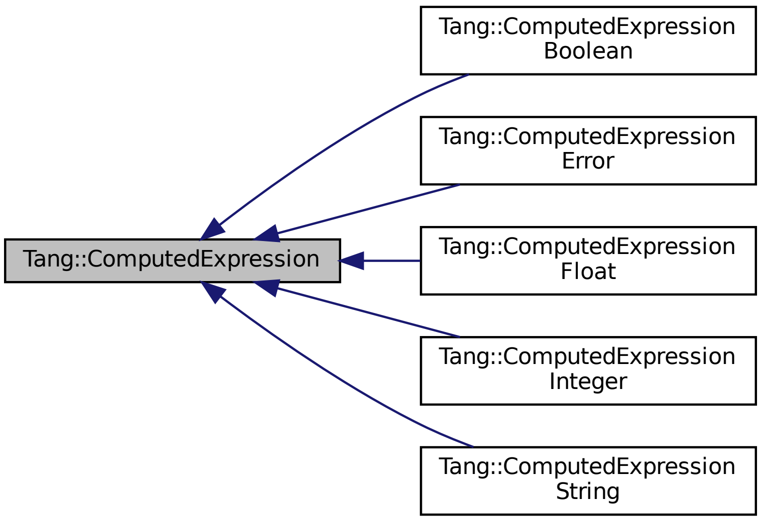
- include/astNodeWhile.hpp
- src/astNodeWhile.cpp

5.18 Tang::ComputedExpression Class Reference

Represents the result of a computation that has been executed.

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

Inheritance diagram for Tang::ComputedExpression:


Tang::ComputedExpression Boolean
Tang::ComputedExpression Error
Tang::ComputedExpression
Tang::ComputedExpression Float
Tang::ComputedExpression Integer
Tang::ComputedExpression String

Public Member Functions

- virtual ∼ComputedExpression ()

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

  Output the contents of the ComputedExpression as a string.
- virtual GarbageCollected makeCopy () const

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

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

- virtual bool is_equal (const double &val) const

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

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

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

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

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

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

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

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

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

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

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

  Compute the logical not of this value.
- virtual GarbageCollected __lessThan (const GarbageCollected &rhs) const

  Compute the "less than" comparison.
- virtual GarbageCollected __equal (const GarbageCollected &rhs) const

  Perform an equalit test.
- virtual GarbageCollected __integer () const

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

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

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

  Perform a type cast to string.

5.18.1 Detailed Description

Represents the result of a computation that has been executed.

By default, it will represent a NULL value.

5.18.2 Member Function Documentation

5.18.2.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

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

5.18.2.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

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

5.18.2.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

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

5.18.2.4 __equal()

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

Perform an equalit test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

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

5.18.2.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

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

5.18.2.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

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

5.18.2.7 __lessThan()

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

Compute the "less than" comparison.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

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

5.18.2.8 __modulo()

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

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

Parameters

rhs	The GarbageCollected value to modulo this by.

Returns

The result of the operation.

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

5.18.2.9 __multiply()

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

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

Parameters

rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

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

5.18.2.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

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

5.18.2.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

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

5.18.2.12 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

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

5.18.2.13 __subtract()

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

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

Parameters

rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

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

5.18.2.14 dump()

```
string ComputedExpression::dump ( ) const  [virtual]
```

Output the contents of the ComputedExpression as a string.

Returns

A string representation of the computed expression.

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

5.18.2.15 is_equal() [1/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.18.2.16 is_equal() [2/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.18.2.17 is_equal() [3/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionError.

5.18.2.18 is_equal() [4/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.18.2.19 is_equal() [5/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

5.18.2.20 is_equal() [6/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionString.

5.18.2.21 makeCopy()

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

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

Returns

A Tang::GarbageCollected value for the new ComputedExpression.

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

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

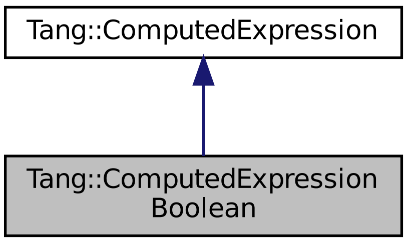
- include/computedExpression.hpp
- src/computedExpression.cpp

5.19 Tang::ComputedExpressionBoolean Class Reference

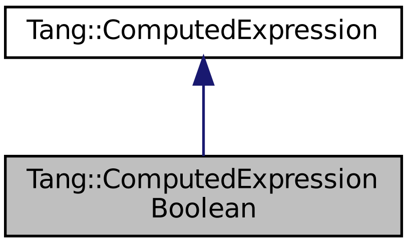
Represents an Boolean that is the result of a computation.

```
#include <computedExpressionBoolean.hpp>
```

Inheritance diagram for Tang::ComputedExpressionBoolean:



Collaboration diagram for Tang::ComputedExpressionBoolean:



Public Member Functions

- ComputedExpressionBoolean (bool val)

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

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

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

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

  Compute the logical not of this value.
- virtual GarbageCollected __equal (const GarbageCollected &rhs) const override

  Perform an equalit test.
- virtual GarbageCollected __integer () const override

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

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

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

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

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

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

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

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

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

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

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

  Compute the result of negating this value.
- virtual GarbageCollected __lessThan (const GarbageCollected &rhs) const

  Compute the "less than" comparison.
- virtual GarbageCollected __string () const

  Perform a type cast to string.

5.19.1 Detailed Description

Represents an Boolean that is the result of a computation.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 ComputedExpressionBoolean()

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

Construct an Boolean result.

Parameters

val	The boolean value.

5.19.3 Member Function Documentation

5.19.3.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

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

5.19.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.19.3.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

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

5.19.3.4 __equal()

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

Perform an equalit test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.19.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.19.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.19.3.7 __lessThan()

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

Compute the "less than" comparison.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

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

5.19.3.8 __modulo()

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

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

Parameters

rhs	The GarbageCollected value to modulo this by.

Returns

The result of the operation.

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

5.19.3.9 __multiply()

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

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

Parameters

rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

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

5.19.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

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

5.19.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.19.3.12 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

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

5.19.3.13 __subtract()

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

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

Parameters

rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

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

5.19.3.14 dump()

```
string ComputedExpressionBoolean::dump ( ) const  [override], [virtual]
```

Output the contents of the ComputedExpression as a string.

Returns

A string representation of the computed expression.

Reimplemented from Tang::ComputedExpression.

5.19.3.15 is_equal() [1/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.19.3.16 is_equal() [2/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.19.3.17 is_equal() [3/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionError.

5.19.3.18 is_equal() [4/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.19.3.19 is_equal() [5/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

5.19.3.20 is_equal() [6/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionString.

5.19.3.21 makeCopy()

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

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

Returns

A Tang::GarbageCollected value for the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

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

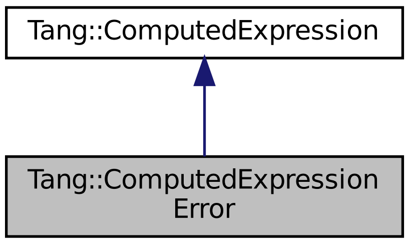
- include/computedExpressionBoolean.hpp
- src/computedExpressionBoolean.cpp

5.20 Tang::ComputedExpressionError Class Reference

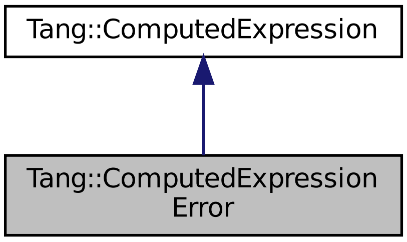
Represents a Runtime Error.

```
#include <computedExpressionError.hpp>
```

Inheritance diagram for Tang::ComputedExpressionError:



Collaboration diagram for Tang::ComputedExpressionError:



Public Member Functions

- ComputedExpressionError (Tang::Error error)

  Construct a Runtime Error.
- virtual std::string dump () const override

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

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

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

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

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

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

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

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

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

  Compute the logical not of this value.
- virtual GarbageCollected __lessThan (const GarbageCollected &rhs) const override

  Compute the "less than" comparison.
- virtual GarbageCollected __equal (const GarbageCollected &rhs) const override

  Perform an equalit test.
- virtual GarbageCollected __integer () const override

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

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

  Perform a type cast to boolean.

- virtual GarbageCollected __string () const override

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

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

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

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

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

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

5.20.1 Detailed Description

Represents a Runtime Error.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 ComputedExpressionError()

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

Construct a Runtime Error.

Parameters

error	The Tang::Error object.

5.20.3 Member Function Documentation

5.20.3.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.4 __equal()

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

Perform an equalit test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.7 __lessThan()

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

Compute the "less than" comparison.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.8 __modulo()

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

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

Parameters

rhs	The GarbageCollected value to modulo this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.9 __multiply()

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

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

Parameters

rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.12 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.13 __subtract()

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

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

Parameters

rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.14 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.

5.20.3.15 is_equal() [1/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.20.3.16 is_equal() [2/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.20.3.17 is_equal() [3/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.20.3.18 is_equal() [4/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.20.3.19 is_equal() [5/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

5.20.3.20 is_equal() [6/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionString.

5.20.3.21 makeCopy()

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

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

Returns

A Tang::GarbageCollected value for the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

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

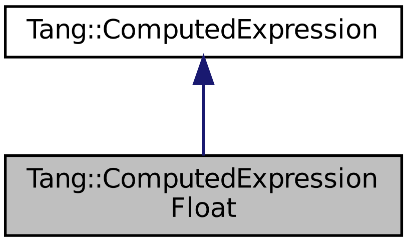
- include/computedExpressionError.hpp
- src/computedExpressionError.cpp

5.21 Tang::ComputedExpressionFloat Class Reference

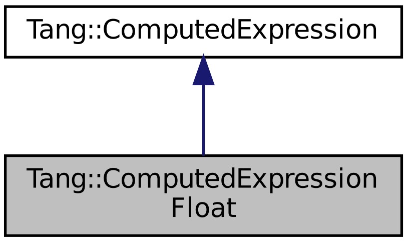
Represents a Float that is the result of a computation.

```
#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.
- GarbageCollected 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 bool is_equal (const bool &val) const override

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

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

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

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

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

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

  Compute the logical not of this value.
- virtual GarbageCollected __lessThan (const GarbageCollected &rhs) const override

  Compute the "less than" comparison.
- virtual GarbageCollected __equal (const GarbageCollected &rhs) const override

  Perform an equalit test.
- virtual GarbageCollected __integer () const override

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

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

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

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

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

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

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

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

Friends

- class **ComputedExpressionInteger**

5.21.1 Detailed Description

Represents a Float that is the result of a computation.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 ComputedExpressionFloat()

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

Construct a Float result.

Parameters

val	The float value.

5.21.3 Member Function Documentation

5.21.3.1 __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.

5.21.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.4 __equal()

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

Perform an equalit test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.7 __lessThan()

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

Compute the "less than" comparison.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.8 __modulo()

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

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

Parameters

rhs	The GarbageCollected value to modulo this by.

Returns

The result of the operation.

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

5.21.3.9 __multiply()

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

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

Parameters

rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.12 __string()

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

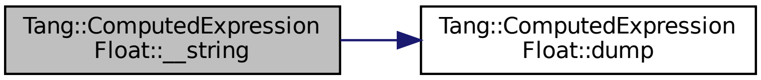
Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Float::__string
Tang::ComputedExpression Float::dump

5.21.3.13 __subtract()

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

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

Parameters

rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.14 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.

5.21.3.15 is_equal() [1/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.21.3.16 is_equal() [2/6]

```
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.

5.21.3.17 is_equal() [3/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionError.

5.21.3.18 is_equal() [4/6]

```
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.

5.21.3.19 is_equal() [5/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

5.21.3.20 is_equal() [6/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionString.

5.21.3.21 makeCopy()

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

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

Returns

A Tang::GarbageCollected value for the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

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

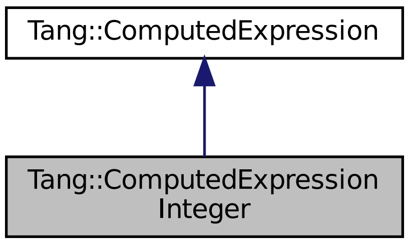
- include/computedExpressionFloat.hpp
- src/computedExpressionFloat.cpp

5.22 Tang::ComputedExpressionInteger Class Reference

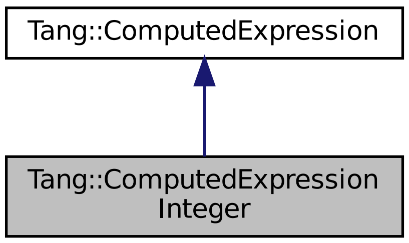
Represents an Integer that is the result of a computation.

```
#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.
- GarbageCollected 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 bool is_equal (const bool &val) const override

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

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

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

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

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

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

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

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

  Compute the logical not of this value.
- virtual GarbageCollected __lessThan (const GarbageCollected &rhs) const override

  Compute the "less than" comparison.
- virtual GarbageCollected __equal (const GarbageCollected &rhs) const override

  Perform an equalit test.
- virtual GarbageCollected __integer () const override

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

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

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

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

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

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

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

Friends

- class **ComputedExpressionFloat**

5.22.1 Detailed Description

Represents an Integer that is the result of a computation.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 ComputedExpressionInteger()

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

Construct an Integer result.

Parameters

val	The integer value.

5.22.3 Member Function Documentation

5.22.3.1 __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.

5.22.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.4 __equal()

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

Perform an equalit test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.7 __lessThan()

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

Compute the "less than" comparison.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.8 __modulo()

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

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

Parameters

rhs	The GarbageCollected value to modulo this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.9 __multiply()

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

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

Parameters

rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.12 __string()

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

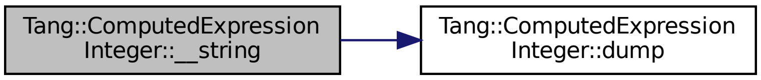
Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Integer::__string
Tang::ComputedExpression Integer::dump

5.22.3.13 __subtract()

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

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

Parameters

rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.14 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.

5.22.3.15 is_equal() [1/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.22.3.16 is_equal() [2/6]

```
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.

5.22.3.17 is_equal() [3/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionError.

5.22.3.18 is_equal() [4/6]

```
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.

5.22.3.19 is_equal() [5/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

5.22.3.20 is_equal() [6/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionString.

5.22.3.21 makeCopy()

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

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

Returns

A Tang::GarbageCollected value for the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

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

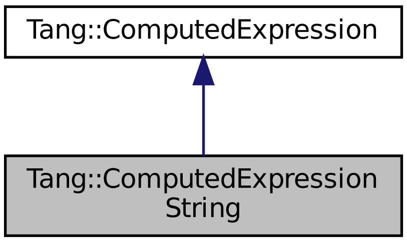
- include/computedExpressionInteger.hpp
- src/computedExpressionInteger.cpp

5.23 Tang::ComputedExpressionString Class Reference

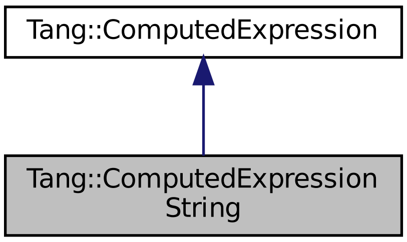
Represents a String that is the result of a computation.

```
#include <computedExpressionString.hpp>
```

Inheritance diagram for Tang::ComputedExpressionString:



Collaboration diagram for Tang::ComputedExpressionString:



Public Member Functions

- ComputedExpressionString (std::string val)

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

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

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

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

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

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

  Compute the logical not of this value.
- virtual GarbageCollected __lessThan (const GarbageCollected &rhs) const override

  Compute the "less than" comparison.

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

    Perform an equalit test.
- virtual GarbageCollected __boolean () const override

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

    Perform a type cast to string.
- virtual bool 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 bool is_equal (const std::nullptr_t &val) const

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

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

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

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

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

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

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

    Perform a type cast to float.

5.23.1 Detailed Description

Represents a String that is the result of a computation.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 ComputedExpressionString()

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

Construct a String result.

Parameters

val	The string value.

5.23.3 Member Function Documentation

5.23.3.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.23.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.23.3.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

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

5.23.3.4 __equal()

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

Perform an equalit test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.23.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

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

5.23.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

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

5.23.3.7 __lessThan()

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

Compute the "less than" comparison.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.23.3.8 __modulo()

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

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

Parameters

rhs	The GarbageCollected value to modulo this by.

Returns

The result of the operation.

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

5.23.3.9 __multiply()

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

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

Parameters

rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

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

5.23.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

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

5.23.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.23.3.12 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.23.3.13 __subtract()

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

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

Parameters

rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

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

5.23.3.14 dump()

```
string ComputedExpressionString::dump ( ) const  [override], [virtual]
```

Output the contents of the ComputedExpression as a string.

Returns

A string representation of the computed expression.

Reimplemented from Tang::ComputedExpression.

5.23.3.15 is_equal() [1/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.23.3.16 is_equal() [2/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.23.3.17 is_equal() [3/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionError.

5.23.3.18 is_equal() [4/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

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

5.23.3.19 is_equal() [5/6]

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

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

5.23.3.20 is_equal() [6/6]

```
bool ComputedExpressionString::is_equal (
             const string & val ) const  [override], [virtual]
```

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

Parameters

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.23.3.21 makeCopy()

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

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

Returns

A Tang::GarbageCollected value for the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

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

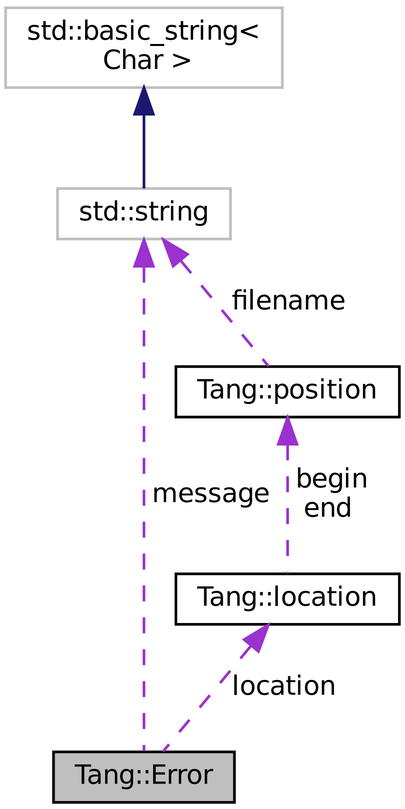
- include/computedExpressionString.hpp
- src/computedExpressionString.cpp

5.24 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.

5.24.1 Detailed Description

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

5.24.2 Constructor & Destructor Documentation

5.24.2.1 Error() [1/2]

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

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

Parameters

message	The error message as a string.

5.24.2.2 Error() [2/2]

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

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

Parameters

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

5.24.3 Friends And Related Function Documentation

5.24.3.1 operator<<

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

Add friendly output.

Parameters

out	The output stream.
error	The Error object.

Returns

The output stream.

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

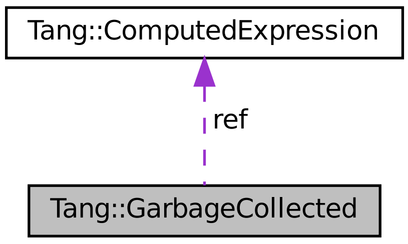
- include/error.hpp
- src/error.cpp

5.25 Tang::GarbageCollected Class Reference

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

```
#include <garbageCollected.hpp>
```

Collaboration diagram for Tang::GarbageCollected:



Public Member Functions

- GarbageCollected (const GarbageCollected &other)

  Copy Constructor.
- GarbageCollected (GarbageCollected &&other)

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

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

  Move Assignment.
- ∼GarbageCollected ()

  Destructor.
- ComputedExpression ∗ operator-> () const

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

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

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

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

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

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

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

  Compare the GarbageCollected tracked object with a supplied value.
- bool operator== (const std::nullptr_t &null) const

  Compare the GarbageCollected tracked object with a supplied value.
- GarbageCollected operator+ (const GarbageCollected &rhs) const

  Perform an addition between two GarbageCollected values.
- GarbageCollected operator- (const GarbageCollected &rhs) const

  Perform a subtraction between two GarbageCollected values.
- GarbageCollected operator∗ (const GarbageCollected &rhs) const

  Perform a multiplication between two GarbageCollected values.
- GarbageCollected operator/ (const GarbageCollected &rhs) const

  Perform a division between two GarbageCollected values.
- GarbageCollected operator% (const GarbageCollected &rhs) const

  Perform a modulo between two GarbageCollected values.
- GarbageCollected operator- () const

  Perform a negation on the GarbageCollected value.
- GarbageCollected operator! () const

  Perform a logical not on the GarbageCollected value.
- GarbageCollected operator< (const GarbageCollected &rhs) const

  Perform a < between two GarbageCollected values.
- GarbageCollected operator<= (const GarbageCollected &rhs) const

  Perform a <= between two GarbageCollected values.
- GarbageCollected operator> (const GarbageCollected &rhs) const

  Perform a > between two GarbageCollected values.
- GarbageCollected operator>= (const GarbageCollected &rhs) const

Perform a >= between two GarbageCollected values.
- GarbageCollected operator== (const GarbageCollected &rhs) const

  Perform a == between two GarbageCollected values.
- GarbageCollected operator!= (const GarbageCollected &rhs) const

  Perform a != between two GarbageCollected values.

Static Public Member Functions

- template<class T , typename... Args>
  static GarbageCollected make (Args... args)

  Creates a garbage-collected object of the specified type.

Protected Member Functions

- GarbageCollected ()

  Constructs a garbage-collected object of the specified type.

Protected Attributes

- size_t ∗ count

  The count of references to the tracked object.
- ComputedExpression ∗ ref

  A reference to the tracked object.
- std::function< void(void)> recycle

  A cleanup function to recycle the object.

Friends

- std::ostream & operator<< (std::ostream &out, const GarbageCollected &gc)

  Add friendly output.

5.25.1 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.

5.25.2 Constructor & Destructor Documentation

5.25.2.1 GarbageCollected() [1/3]

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

Copy Constructor.

Parameters

The	other GarbageCollected object to copy.

5.25.2.2 GarbageCollected() [2/3]

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

Move Constructor.

Parameters

The	other GarbageCollected object to move.

5.25.2.3 ∼GarbageCollected()

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

Destructor.

Clean up the tracked object, if appropriate.

5.25.2.4 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.

5.25.3 Member Function Documentation

5.25.3.1 make()

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

Creates a garbage-collected object of the specified type.

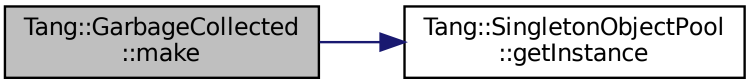
Parameters

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

Returns

A GarbageCollected object.

Here is the call graph for this function:


Tang::GarbageCollected ::make
Tang::SingletonObjectPool ::getInstance

5.25.3.2 operator"!()

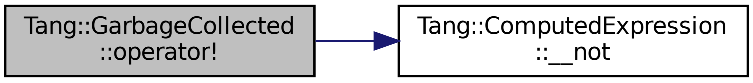
```
GarbageCollected GarbageCollected::operator!  ( ) const
```

Perform a logical not on the GarbageCollected value.

Returns

The result of the operation.

Here is the call graph for this function:


Tang::GarbageCollected ::operator!
Tang::ComputedExpression ::__not

5.25.3.3 operator"!=()

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

Perform a != between two GarbageCollected values.

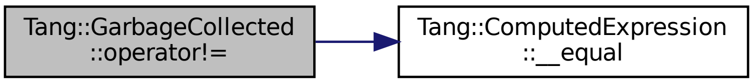
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:


Tang::GarbageCollected ::operator!=
Tang::ComputedExpression ::__equal

5.25.3.4 operator%()

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

Perform a modulo between two GarbageCollected values.

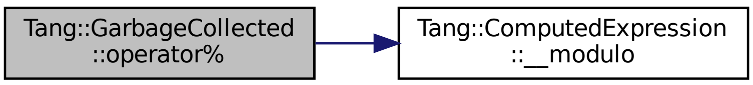
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:


Tang::GarbageCollected ::operator%
Tang::ComputedExpression ::__modulo

5.25.3.5 operator*() [1/2]

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

Access the tracked object.

Returns

A reference to the tracked object.

5.25.3.6 operator*() [2/2]

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

Perform a multiplication between two GarbageCollected values.

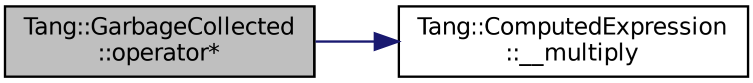
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:


Tang::GarbageCollected ::operator*
Tang::ComputedExpression ::__multiply

5.25.3.7 operator+()

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

Perform an addition between two GarbageCollected values.

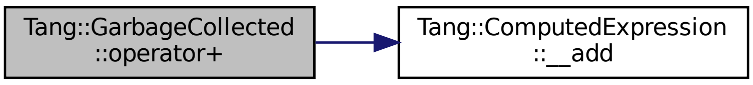
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:


Tang::GarbageCollected ::operator+
Tang::ComputedExpression ::__add

5.25.3.8 operator-() [1/2]

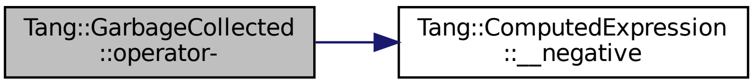
```
GarbageCollected GarbageCollected::operator- ( ) const
```

Perform a negation on the GarbageCollected value.

Returns

The result of the operation.

Here is the call graph for this function:


Tang::GarbageCollected ::operator-
Tang::ComputedExpression ::__negative

5.25.3.9 operator-() [2/2]

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

Perform a subtraction between two GarbageCollected values.

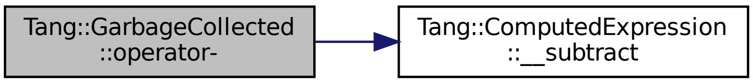
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:


Tang::GarbageCollected ::operator-
Tang::ComputedExpression ::__subtract

5.25.3.10 operator->()

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

Access the tracked object as a pointer.

Returns

A pointer to the tracked object.

5.25.3.11 operator/()

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

Perform a division between two GarbageCollected values.

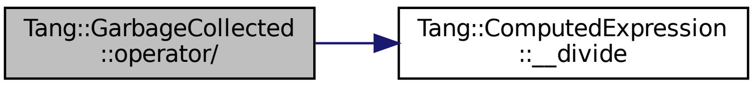
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:


Tang::GarbageCollected ::operator/
Tang::ComputedExpression ::__divide

5.25.3.12 operator<()

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

Perform a < between two GarbageCollected values.

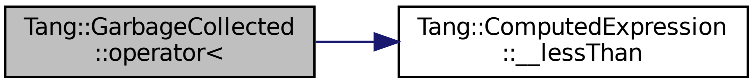
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:


Tang::GarbageCollected ::operator<
Tang::ComputedExpression ::__lessThan

5.25.3.13 operator<=()

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

Perform a <= between two GarbageCollected values.

Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

5.25.3.14 operator=() `[1/2]`

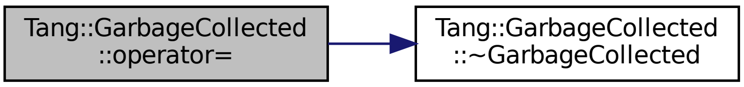
```
GarbageCollected& Tang::GarbageCollected::operator= (
            const GarbageCollected & other ) [inline]
```

Copy Assignment.

Parameters

The	other GarbageCollected object.

Here is the call graph for this function:


Tang::GarbageCollected ::operator=
Tang::GarbageCollected ::~GarbageCollected

5.25.3.15 operator=() `[2/2]`

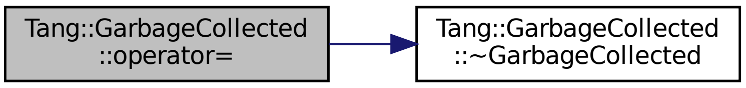
```
GarbageCollected& Tang::GarbageCollected::operator= (
            GarbageCollected && other ) [inline]
```

Move Assignment.

Parameters

The	other GarbageCollected object.

Here is the call graph for this function:


Tang::GarbageCollected ::operator=
Tang::GarbageCollected ::~GarbageCollected

5.25.3.16 operator==() `[1/8]`

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

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.25.3.17 operator==() `[2/8]`

```
bool GarbageCollected::operator== (
             const char *const & val ) const
```

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.25.3.18 operator==() [3/8]

```
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.

5.25.3.19 operator==() [4/8]

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

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.25.3.20 operator==() [5/8]

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

Perform a == between two GarbageCollected values.

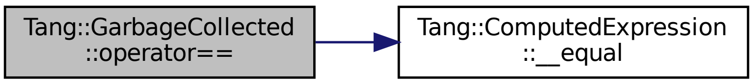
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:


Tang::GarbageCollected ::operator==
Tang::ComputedExpression ::__equal

5.25.3.21 operator==() [6/8]

```
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.

5.25.3.22 operator==() [7/8]

```
bool GarbageCollected::operator== (
             const std::nullptr_t & null ) const
```

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.25.3.23 operator==() [8/8]

```
bool GarbageCollected::operator== (
            const std::string & val ) const
```

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.25.3.24 operator>()

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

Perform a > between two GarbageCollected values.

Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

5.25.3.25 operator>=()

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

Perform a >= between two GarbageCollected values.

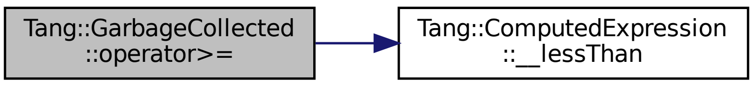
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:


Tang::GarbageCollected ::operator>=
Tang::ComputedExpression ::__lessThan

5.25.4 Friends And Related Function Documentation

5.25.4.1 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:

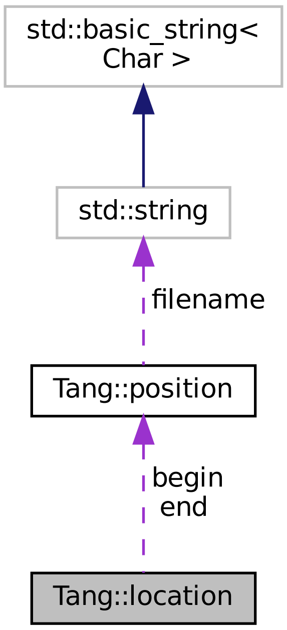
- include/garbageCollected.hpp
- src/garbageCollected.cpp

5.26 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.

5.26.1 Detailed Description

Two points in a source file.

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

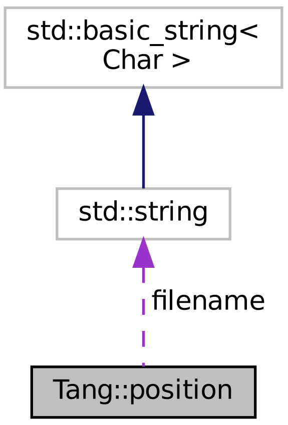
- build/generated/location.hh

5.27 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.

5.27.1 Detailed Description

A point in a source file.

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

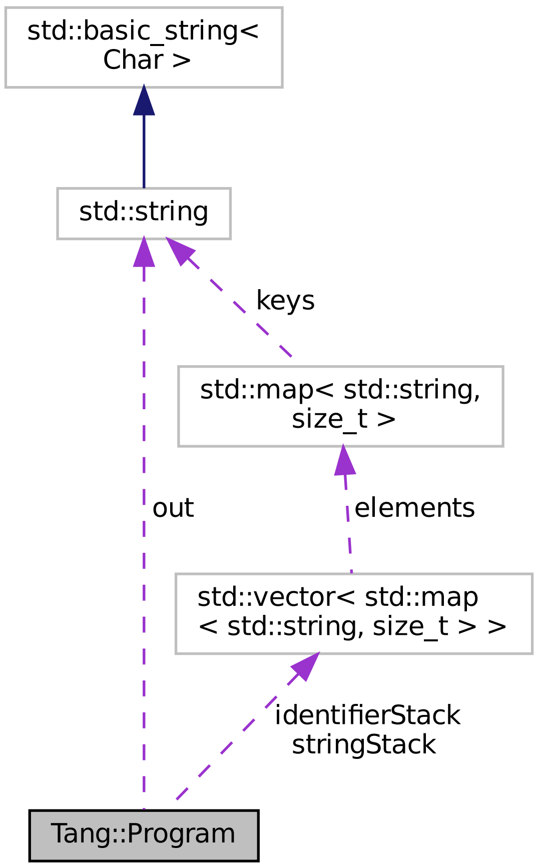
- build/generated/location.hh

5.28 Tang::Program Class Reference

Represents a compiled script or template that may be executed.

```
#include <program.hpp>
```

Collaboration diagram for Tang::Program:


std::basic_string< Char >
std::string
keys
std::map< std::string, size_t >
out
elements
std::vector< std::map < std::string, size_t > >
identifierStack
stringStack
Tang::Program

Public Types

- enum CodeType { Script , Template }

  Indicate the type of code that was supplied to the Program.

Public Member Functions

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

  Create a compiled program using the provided code.
- std::string getCode () const

  Get the code that was provided when the Program was created.
- std::optional< const std::shared_ptr< AstNode > > getAst () const

  Get the AST that was generated by the parser.
- std::string dumpBytecode () const

  Get the Opcodes of the compiled program, formatted like Assembly.
- std::optional< const GarbageCollected > getResult () const

Get the result of the Program execution, if it exists.
- size_t addBytecode (uint64_t)

  Add a uint64_t to the Bytecode.
- const Bytecode & getBytecode ()

  Get the Bytecode vector.
- Program & execute ()

  Execute the program's Bytecode, and return the current Program object.
- bool setJumpTarget (size_t opcodeAddress, uint64_t jumpTarget)

  Set the target address of a Jump opcode.

Public Attributes

- std::string out

  The output of the program, resulting from the program execution.
- std::vector< std::map< std::string, size_t > > identifierStack

  Stack of mappings of identifiers to their stack locations.
- std::vector< std::map< std::string, size_t > > stringStack

  Stack of mappings of strings to their stack locations.

5.28.1 Detailed Description

Represents a compiled script or template that may be executed.

5.28.2 Member Enumeration Documentation

5.28.2.1 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.

5.28.3 Constructor & Destructor Documentation

5.28.3.1 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`.

5.28.4 Member Function Documentation

5.28.4.1 addBytecode()

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

Add a uint64_t to the Bytecode.

Parameters

op	The value to add to the Bytecode.

Returns

The size of the bytecode structure.

5.28.4.2 dumpBytecode()

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

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

Returns

A string containing the Opcode representation.

5.28.4.3 execute()

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

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

Returns

The current Program object.

5.28.4.4 getAst()

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

Get the AST that was generated by the parser.

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

Returns

A pointer to the AST, if it exists.

5.28.4.5 getBytecode()

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

Get the Bytecode vector.

Returns

The Bytecode vector.

5.28.4.6 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.

5.28.4.7 getResult()

```
optional< const GarbageCollected > Program::getResult ( ) const
```

Get the result of the Program execution, if it exists.

Returns

The result of the Program execution, if it exists.

5.28.4.8 setJumpTarget()

```
bool Program::setJumpTarget (
            size_t opcodeAddress,
            uint64_t jumpTarget )
```

Set the target address of a Jump opcode.

Parameters

opcodeAddress	The location of the jump statement.
jumpTarget	The address to jump to.

Returns

Whether or not the jumpTarget was set.

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

5.29 Tang::SingletonObjectPool< T > Class Template Reference

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

```
#include <singletonObjectPool.hpp>
```

Public Member Functions

- T ∗ get ()

  Request an uninitialized memory location from the pool for an object T.
- void recycle (T ∗obj)

  Recycle a memory location for an object T.
- ∼SingletonObjectPool ()

  Destructor.

Static Public Member Functions

- static SingletonObjectPool< T > & getInstance ()
  Get the singleton instance of the object pool.

5.29.1 Detailed Description

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

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

5.29.2 Member Function Documentation

5.29.2.1 get()

```
template<class T >
T* Tang::SingletonObjectPool< T >::get ( )  [inline]
```

Request an uninitialized memory location from the pool for an object T.

Returns

An uninitialized memory location for an object T.

5.29.2.2 getInstance()

```
template<class T >
static SingletonObjectPool<T>& Tang::SingletonObjectPool< T >::getInstance ( )  [inline],
[static]
```

Get the singleton instance of the object pool.

Returns

The singleton instance of the object pool.

5.29.2.3 recycle()

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

Recycle a memory location for an object T.

Parameters

obj	The memory location to recycle.

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

- include/singletonObjectPool.hpp

5.30 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.

5.30.1 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).

5.30.2 Constructor & Destructor Documentation

5.30.2.1 TangBase()

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

The constructor.

Isn't it glorious.

5.30.3 Member Function Documentation

5.30.3.1 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:

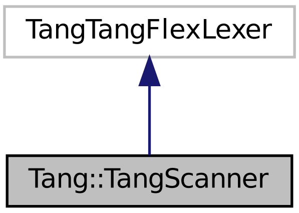
- include/tangBase.hpp
- src/tangBase.cpp

5.31 Tang::TangScanner Class Reference

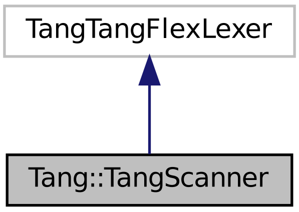
The Flex lexer class for the main Tang language.

```
#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.*

5.31.1 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.

5.31.2 Constructor & Destructor Documentation

5.31.2.1 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)

5.31.3 Member Function Documentation

5.31.3.1 get_next_token()

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

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

Returns

A Bison 3 token representing the lexeme that was recognized.

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

- include/tangScanner.hpp

Chapter 6

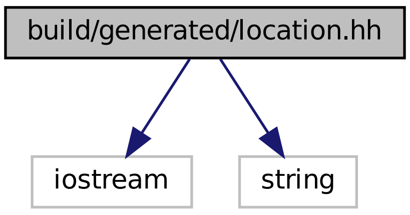
File Documentation

6.1 build/generated/location.hh File Reference

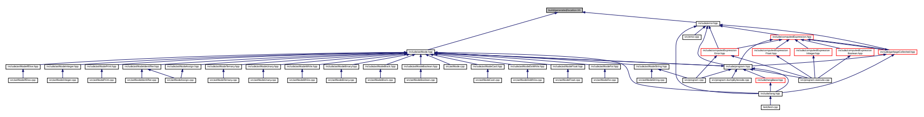
Define the Tang ::location class.

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

Include dependency graph for location.hh:



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



Classes

- class Tang::position

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

  Two points in a source file.

Macros

- #define **YY_NULLPTR** ((void∗)0)

Functions

- position & Tang::operator+= (position &res, position::counter_type width)
  Add width columns, in place.
- position Tang::operator+ (position res, position::counter_type width)
  Add width columns.
- position & Tang::operator-= (position &res, position::counter_type width)
  Subtract width columns, in place.
- position Tang::operator- (position res, position::counter_type width)
  Subtract width columns.
- template<typename YYChar >
  std::basic_ostream< YYChar > & Tang::operator<< (std::basic_ostream< YYChar > &ostr, const position &pos)
  Intercept output stream redirection.
- location & Tang::operator+= (location &res, const location &end)
  Join two locations, in place.
- location Tang::operator+ (location res, const location &end)
  Join two locations.
- location & Tang::operator+= (location &res, location::counter_type width)
  Add width columns to the end position, in place.
- location Tang::operator+ (location res, location::counter_type width)
  Add width columns to the end position.
- location & Tang::operator-= (location &res, location::counter_type width)
  Subtract width columns to the end position, in place.
- location Tang::operator- (location res, location::counter_type width)
  Subtract width columns to the end position.
- template<typename YYChar >
  std::basic_ostream< YYChar > & Tang::operator<< (std::basic_ostream< YYChar > &ostr, const location &loc)
  Intercept output stream redirection.

6.1.1 Detailed Description

Define the Tang ::location class.

6.1.2 Function Documentation

6.1.2.1 operator<<() [1/2]

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

Intercept output stream redirection.

Parameters

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

Avoid duplicate information.

6.1.2.2 operator<<() [2/2]

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

Intercept output stream redirection.

Parameters

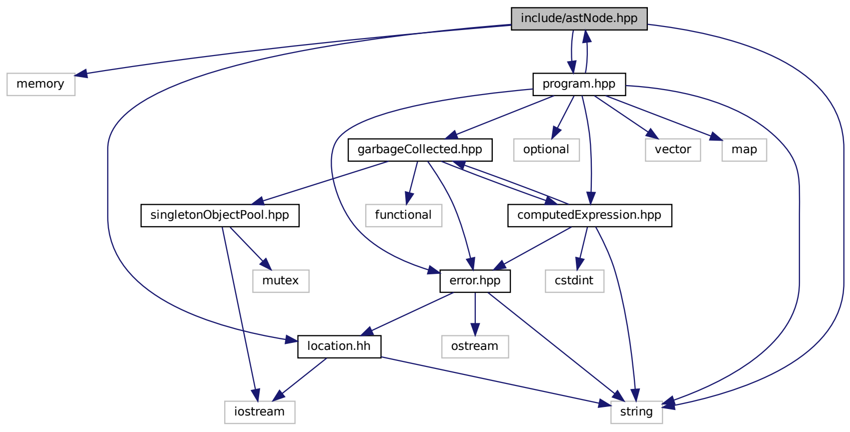
ostr	the destination output stream
pos	a reference to the position to redirect

6.2 include/astNode.hpp File Reference

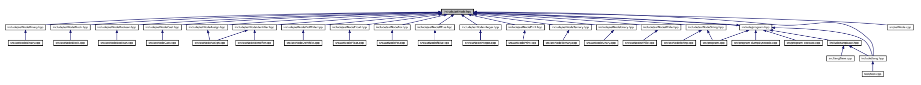
Declare the Tang::AstNode base class.

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

Include dependency graph for astNode.hpp:



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



Classes

- class Tang::AstNode

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

6.2.1 Detailed Description

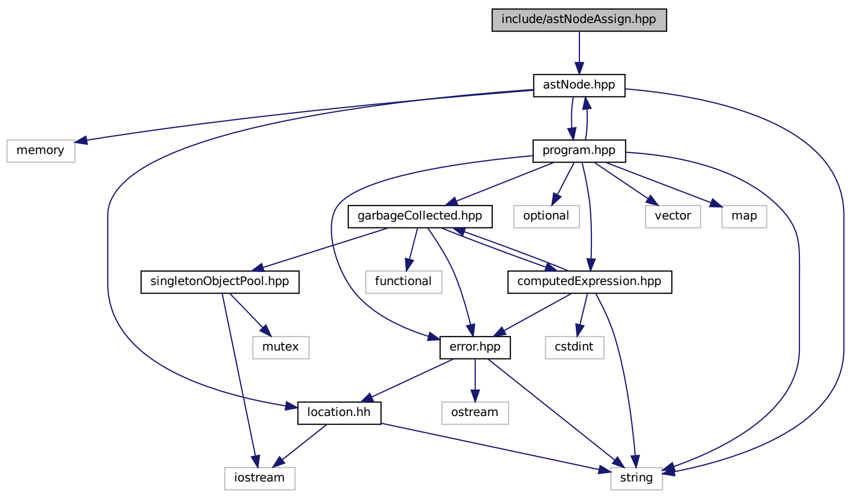
Declare the Tang::AstNode base class.

6.3 include/astNodeAssign.hpp File Reference

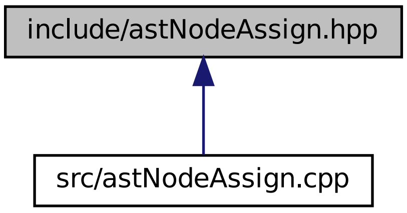
Declare the Tang::AstNodeAssign class.

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

Include dependency graph for astNodeAssign.hpp:



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



Classes

- class Tang::AstNodeAssign

    An AstNode that represents a binary expression.

6.3.1 Detailed Description

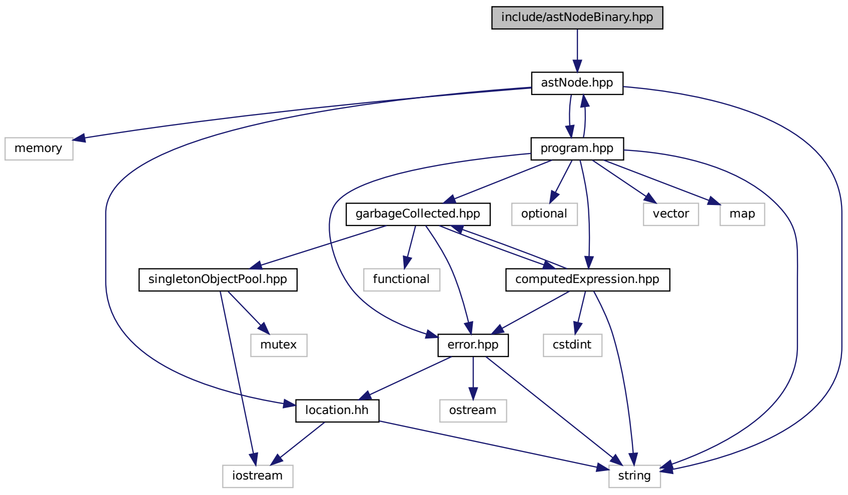
Declare the Tang::AstNodeAssign class.

6.4 include/astNodeBinary.hpp File Reference

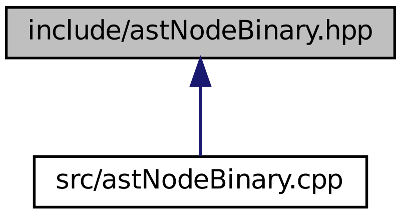
Declare the Tang::AstNodeBinary class.

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

Include dependency graph for astNodeBinary.hpp:



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



Classes

- class Tang::AstNodeBinary

    An AstNode that represents a binary expression.

6.4.1 Detailed Description

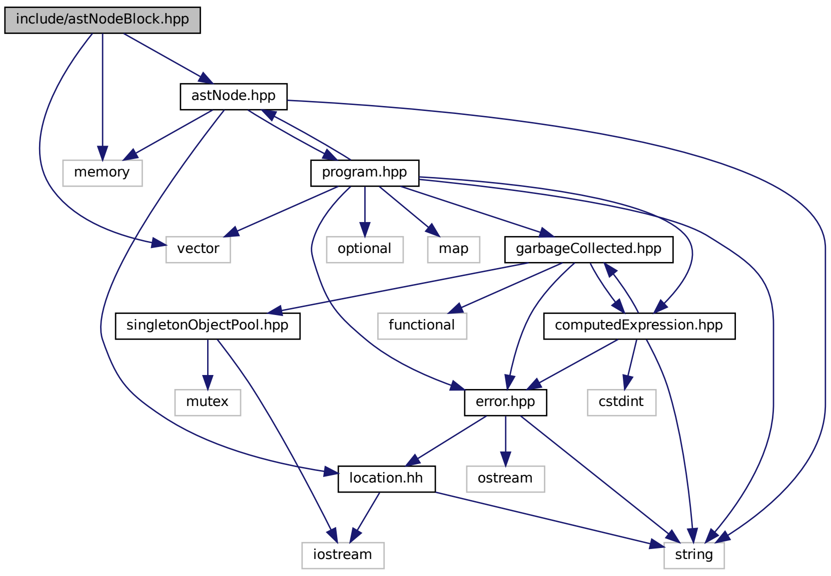
Declare the Tang::AstNodeBinary class.

6.5 include/astNodeBlock.hpp File Reference

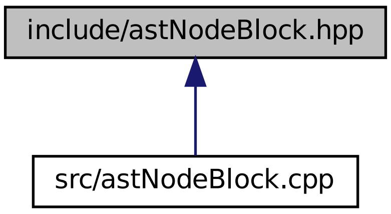
Declare the Tang::AstNodeBlock class.

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

Include dependency graph for astNodeBlock.hpp:



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



Classes

- class Tang::AstNodeBlock

  An AstNode that represents a code block.

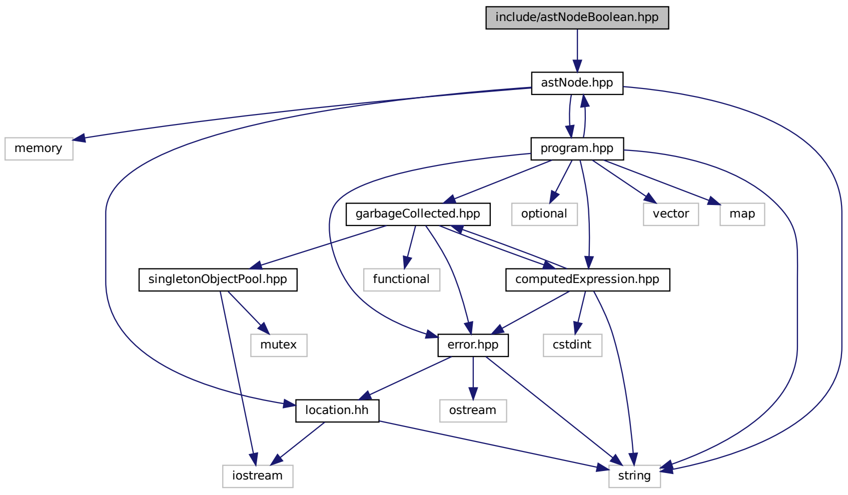
6.5.1 Detailed Description

Declare the Tang::AstNodeBlock class.

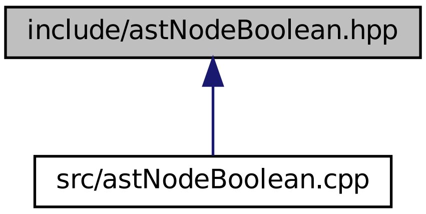
6.6 include/astNodeBoolean.hpp File Reference

Declare the Tang::AstNodeBoolean class.

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



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



Classes

- class Tang::AstNodeBoolean

  An AstNode that represents a boolean literal.

6.6.1 Detailed Description

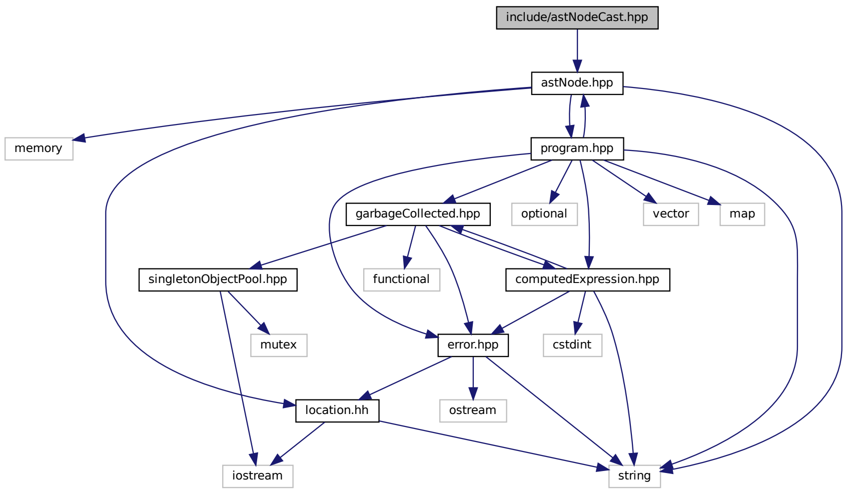
Declare the Tang::AstNodeBoolean class.

6.7 include/astNodeCast.hpp File Reference

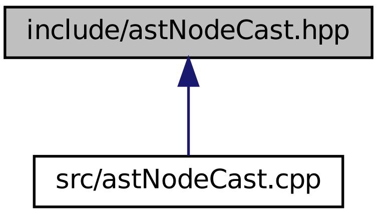
Declare the Tang::AstNodeCast class.

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

Include dependency graph for astNodeCast.hpp:



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



Classes

- class Tang::AstNodeCast

  An AstNode that represents a typecast of an expression.

6.7.1 Detailed Description

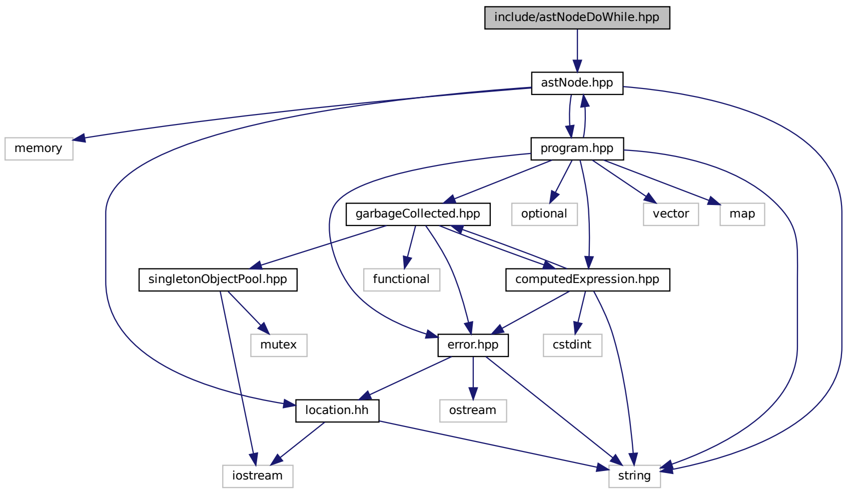
Declare the Tang::AstNodeCast class.

6.8 include/astNodeDoWhile.hpp File Reference

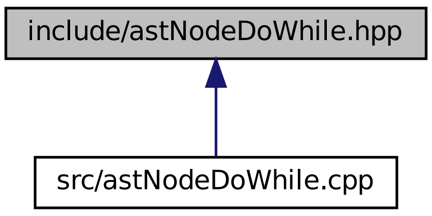
Declare the Tang::AstNodeDoWhile class.

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

Include dependency graph for astNodeDoWhile.hpp:



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



Classes

- class Tang::AstNodeDoWhile

  An AstNode that represents a do..while statement.

6.8.1 Detailed Description

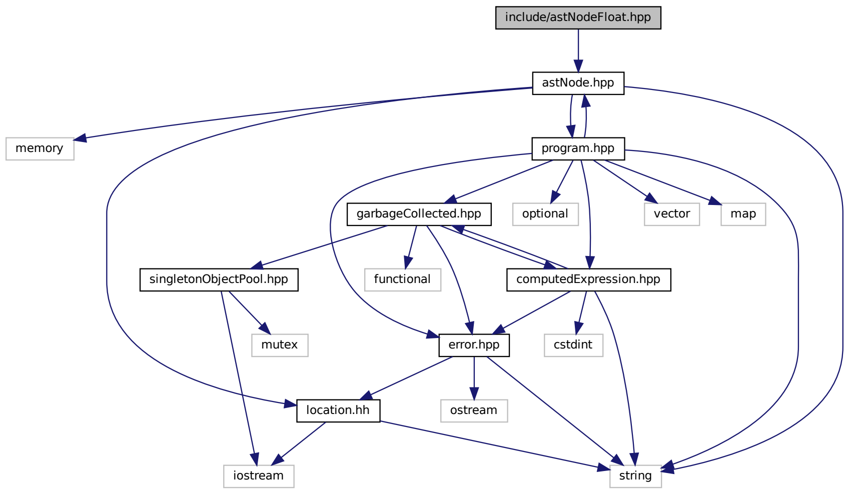
Declare the Tang::AstNodeDoWhile class.

6.9 include/astNodeFloat.hpp File Reference

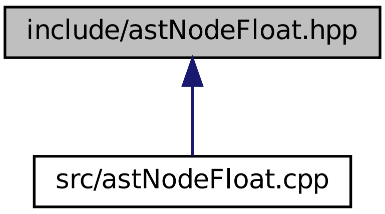
Declare the Tang::AstNodeFloat class.

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

Include dependency graph for astNodeFloat.hpp:



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



Classes

- class Tang::AstNodeFloat

  An AstNode that represents an float literal.

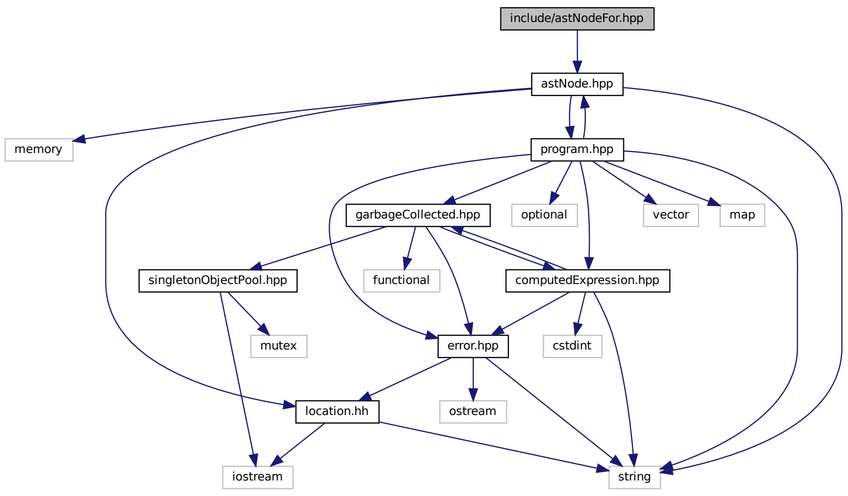
6.9.1 Detailed Description

Declare the Tang::AstNodeFloat class.

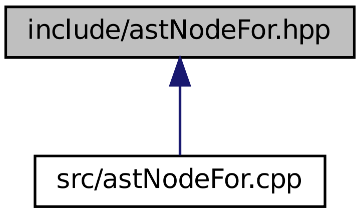
6.10 include/astNodeFor.hpp File Reference

Declare the Tang::AstNodeFor class.

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



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



Classes

- class Tang::AstNodeFor

  An AstNode that represents an if() statement.

6.10.1 Detailed Description

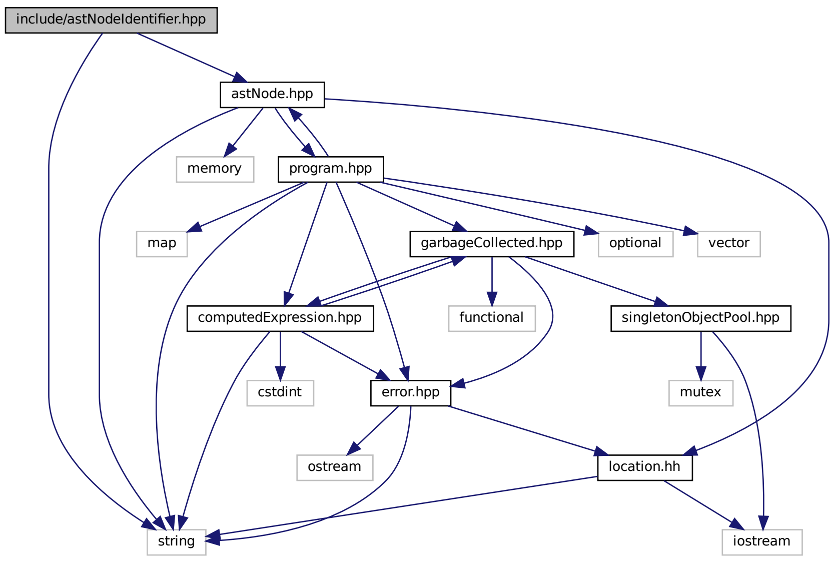
Declare the Tang::AstNodeFor class.

6.11 include/astNodeIdentifier.hpp File Reference

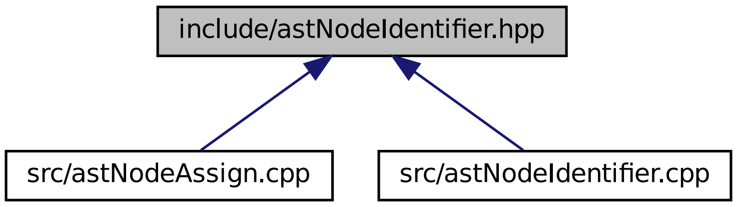
Declare the Tang::AstNodeIdentifier class.

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

Include dependency graph for astNodeIdentifier.hpp:



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



Classes

- class Tang::AstNodeIdentifier

    An AstNode that represents an identifier.

6.11.1 Detailed Description

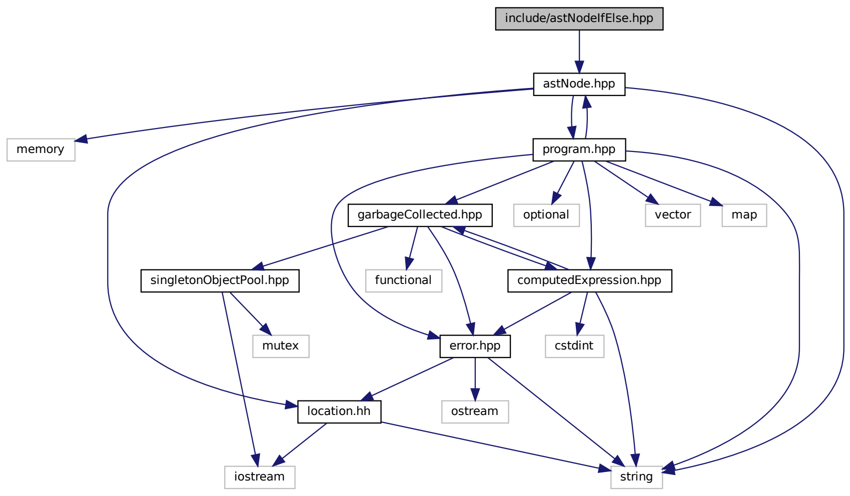
Declare the Tang::AstNodeIdentifier class.

6.12 include/astNodeIfElse.hpp File Reference

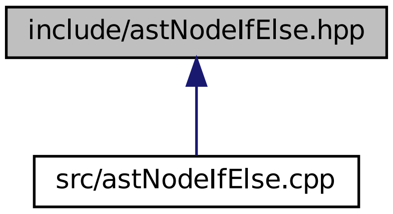
Declare the Tang::AstNodeIfElse class.

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

Include dependency graph for astNodeIfElse.hpp:



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



Classes

- class Tang::AstNodeIfElse

  An AstNode that represents an if..else statement.

6.12.1 Detailed Description

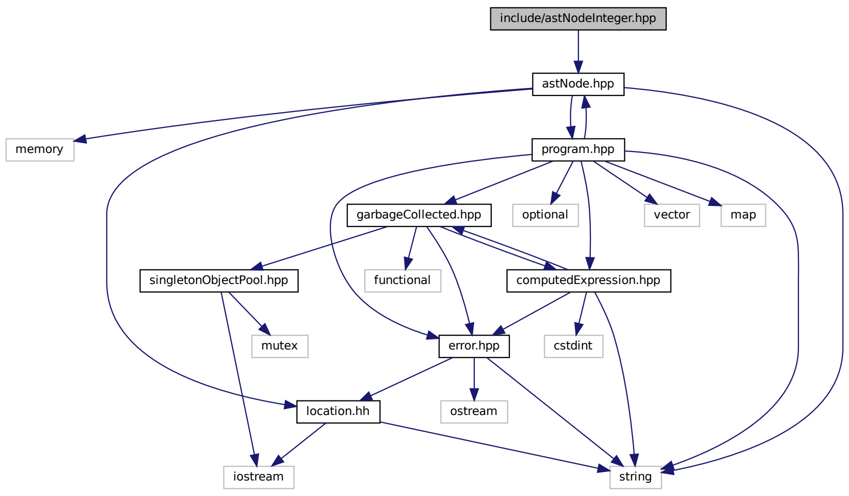
Declare the Tang::AstNodeIfElse class.

6.13 include/astNodeInteger.hpp File Reference

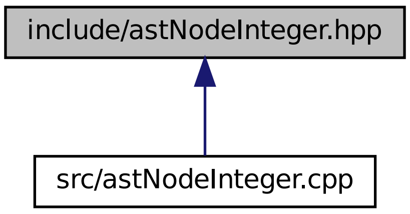
Declare the Tang::AstNodeInteger class.

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

Include dependency graph for astNodeInteger.hpp:



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



Classes

- class Tang::AstNodeInteger

  An AstNode that represents an integer literal.

6.13.1 Detailed Description

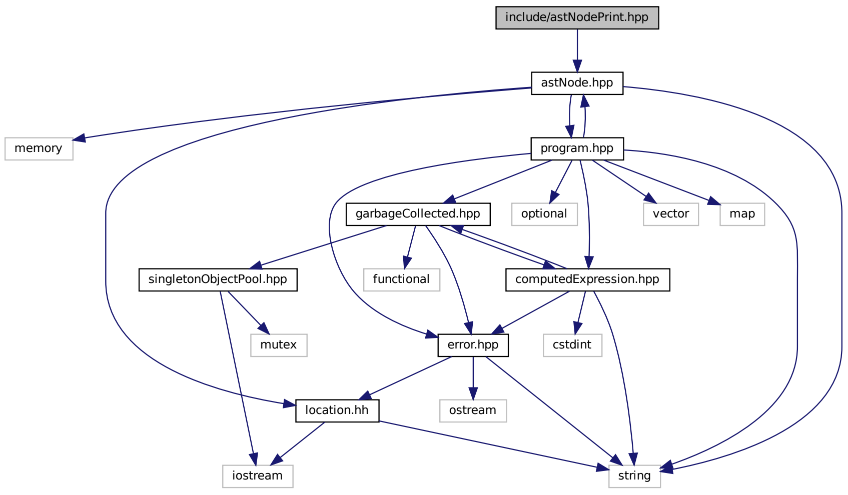
Declare the Tang::AstNodeInteger class.

6.14 include/astNodePrint.hpp File Reference

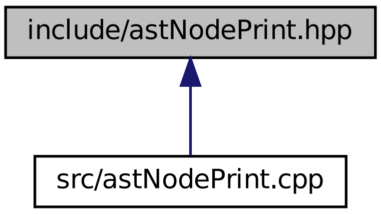
Declare the Tang::AstNodePrint class.

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

Include dependency graph for astNodePrint.hpp:



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



Classes

- class Tang::AstNodePrint

  An AstNode that represents a print typeeration.

6.14.1 Detailed Description

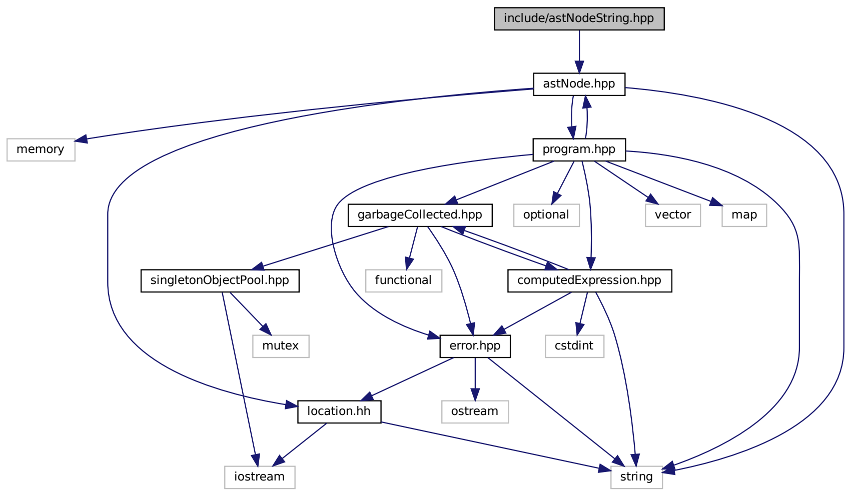
Declare the Tang::AstNodePrint class.

6.15 include/astNodeString.hpp File Reference

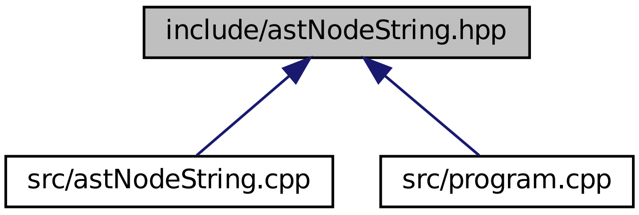
Declare the Tang::AstNodeString class.

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

Include dependency graph for astNodeString.hpp:



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



Classes

- class Tang::AstNodeString

  An AstNode that represents a string literal.

6.15.1 Detailed Description

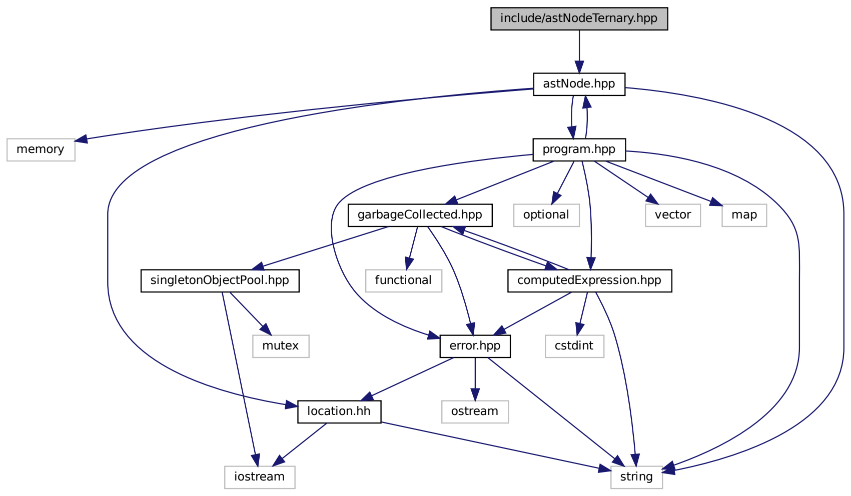
Declare the Tang::AstNodeString class.

6.16 include/astNodeTernary.hpp File Reference

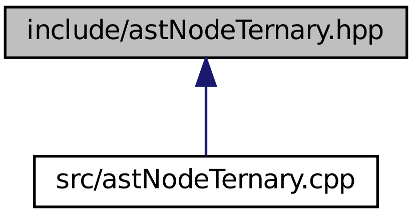
Declare the Tang::AstNodeTernary class.

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

Include dependency graph for astNodeTernary.hpp:



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



Classes

- class Tang::AstNodeTernary

  An AstNode that represents a ternary expression.

6.16.1 Detailed Description

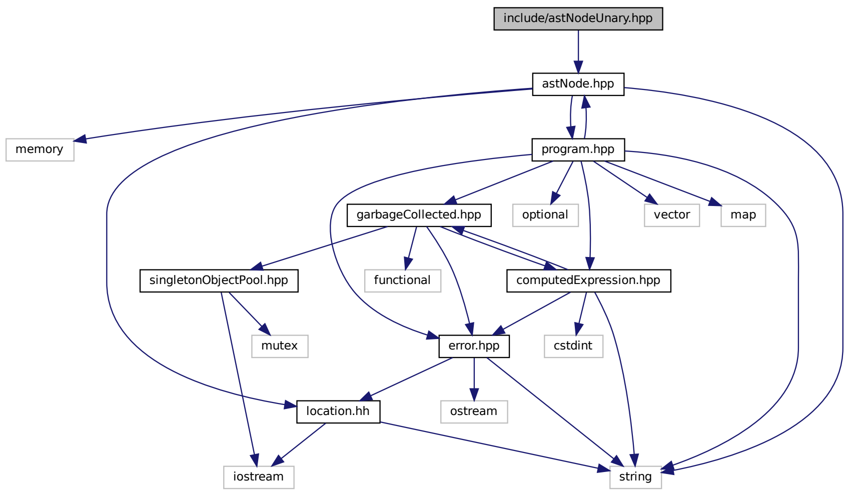
Declare the Tang::AstNodeTernary class.

6.17 include/astNodeUnary.hpp File Reference

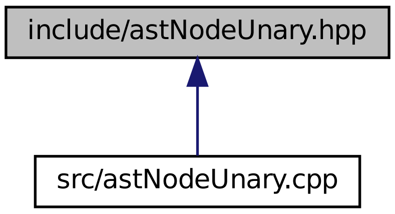
Declare the Tang::AstNodeUnary class.

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

Include dependency graph for astNodeUnary.hpp:



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



Classes

- class Tang::AstNodeUnary

  An AstNode that represents a unary negation.

6.17.1 Detailed Description

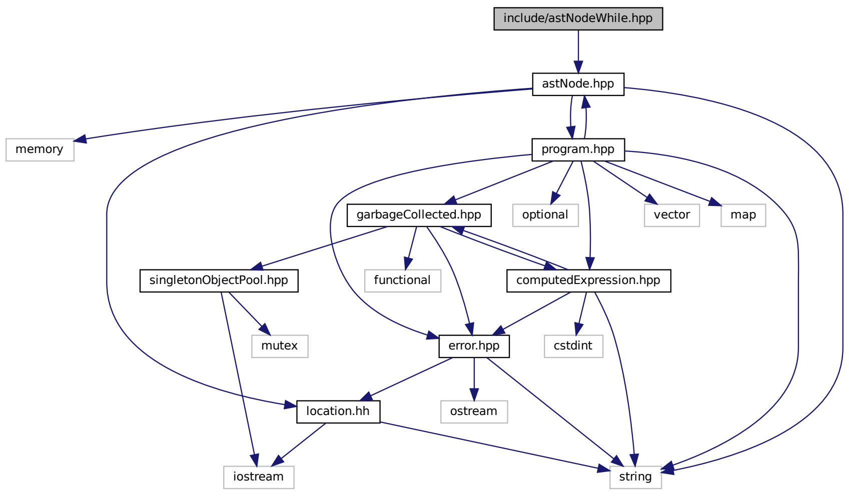
Declare the Tang::AstNodeUnary class.

6.18 include/astNodeWhile.hpp File Reference

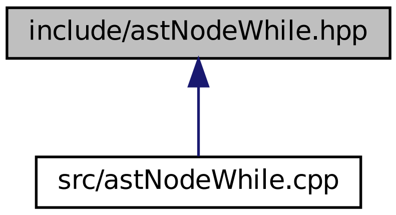
Declare the Tang::AstNodeWhile class.

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

Include dependency graph for astNodeWhile.hpp:



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



Classes

- class Tang::AstNodeWhile

  An AstNode that represents a while statement.

6.18.1 Detailed Description

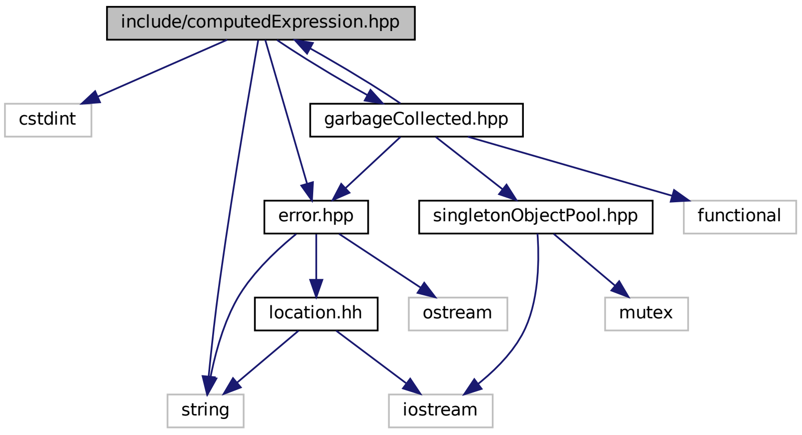
Declare the Tang::AstNodeWhile class.

6.19 include/computedExpression.hpp File Reference

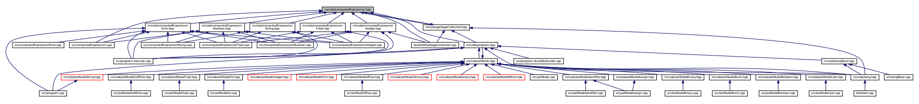
Declare the Tang::ComputedExpression base class.

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

Include dependency graph for computedExpression.hpp:



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



Classes

- class Tang::ComputedExpression

  Represents the result of a computation that has been executed.

6.19.1 Detailed Description

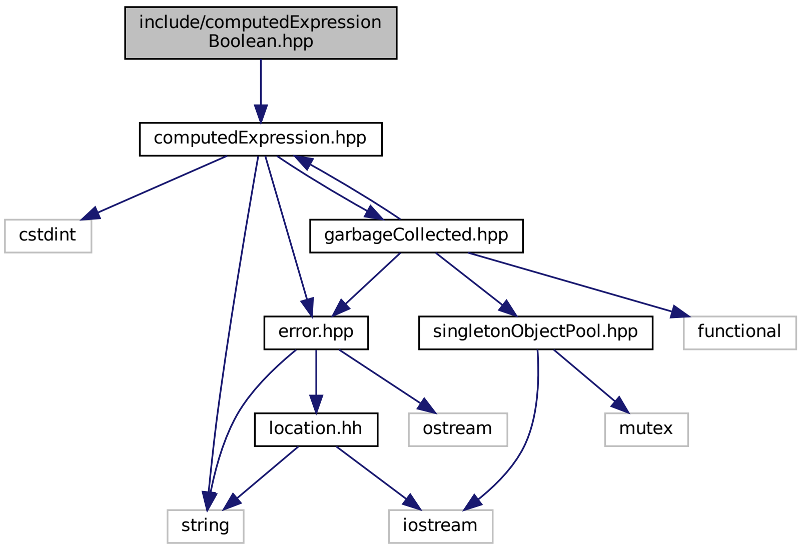
Declare the Tang::ComputedExpression base class.

6.20 include/computedExpressionBoolean.hpp File Reference

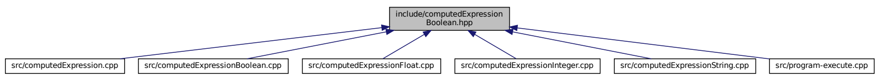
Declare the Tang::ComputedExpressionBoolean class.

`#include "computedExpression.hpp"`

Include dependency graph for computedExpressionBoolean.hpp:



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


include/computedExpression Boolean.hpp
src/computedExpression.cpp
src/computedExpressionBoolean.cpp
src/computedExpressionFloat.cpp
src/computedExpressionInteger.cpp
src/computedExpressionString.cpp
src/program-execute.cpp

Classes

- class Tang::ComputedExpressionBoolean

  Represents an Boolean that is the result of a computation.

6.20.1 Detailed Description

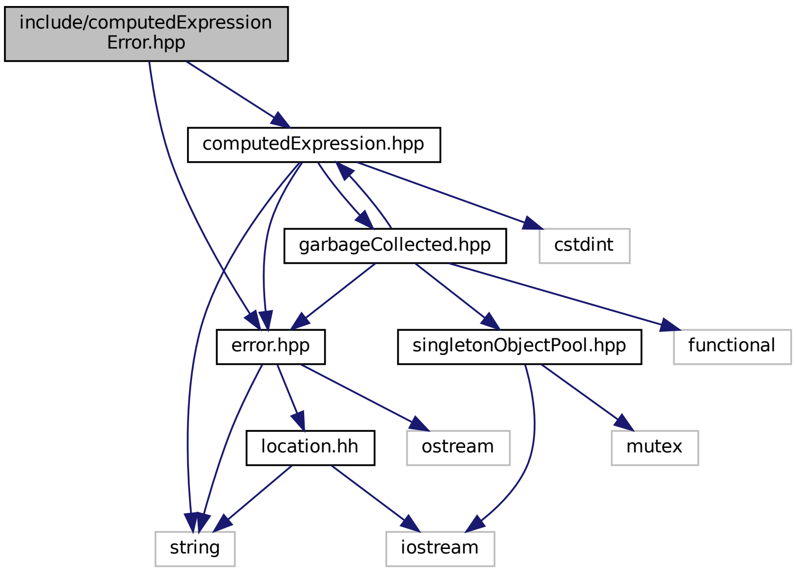
Declare the Tang::ComputedExpressionBoolean class.

6.21 include/computedExpressionError.hpp File Reference

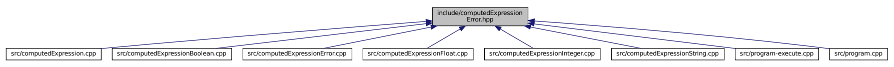
Declare the Tang::ComputedExpressionError class.

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

Include dependency graph for computedExpressionError.hpp:



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


include/computedExpression Error.hpp
src/computedExpression.cpp
src/computedExpressionBoolean.cpp
src/computedExpressionError.cpp
src/computedExpressionFloat.cpp
src/computedExpressionInteger.cpp
src/computedExpressionString.cpp
src/program-execute.cpp
src/program.cpp

Classes

- class Tang::ComputedExpressionError

  Represents a Runtime Error.

6.21.1 Detailed Description

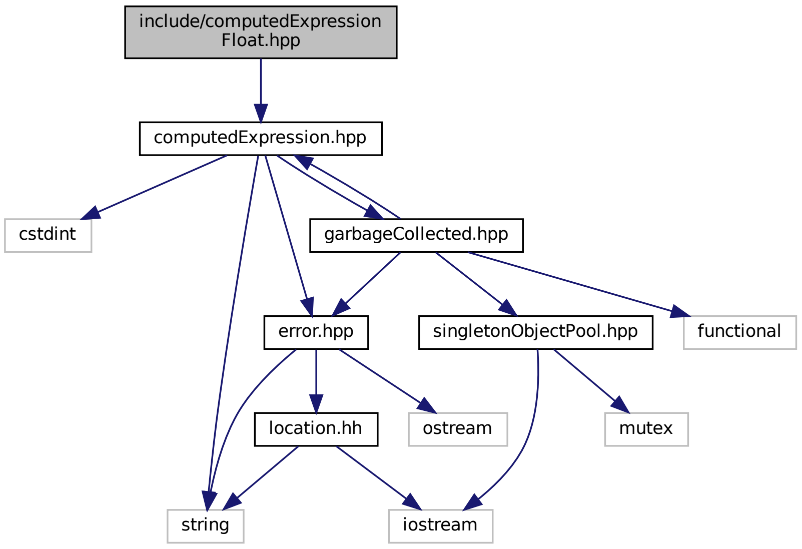
Declare the Tang::ComputedExpressionError class.

6.22 include/computedExpressionFloat.hpp File Reference

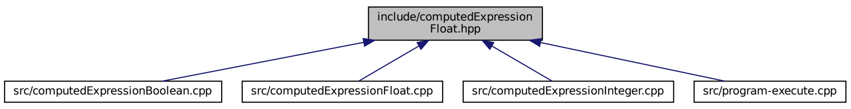
Declare the Tang::ComputedExpressionFloat class.

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

Include dependency graph for computedExpressionFloat.hpp:



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


include/computedExpression Float.hpp
src/computedExpressionBoolean.cpp
src/computedExpressionFloat.cpp
src/computedExpressionInteger.cpp
src/program-execute.cpp

Classes

- class Tang::ComputedExpressionFloat

  Represents a Float that is the result of a computation.

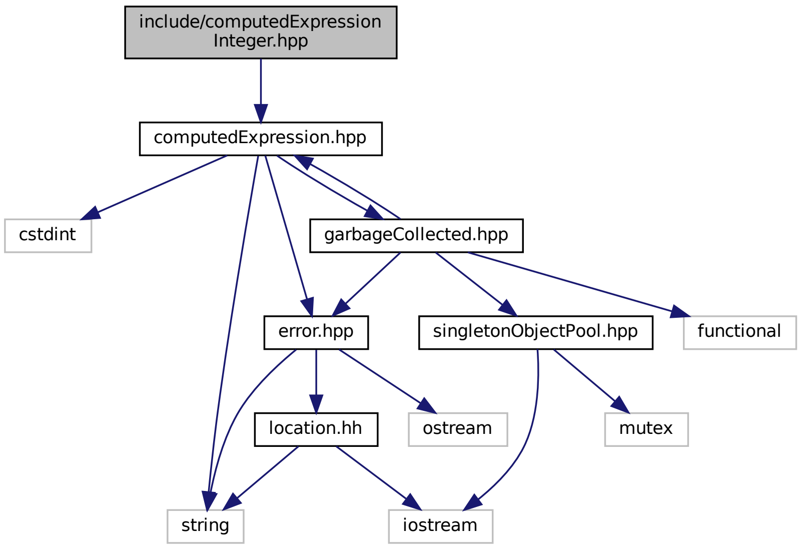
6.22.1 Detailed Description

Declare the Tang::ComputedExpressionFloat class.

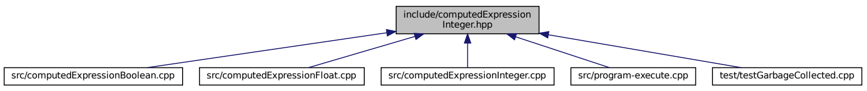
6.23 include/computedExpressionInteger.hpp File Reference

Declare the Tang::ComputedExpressionInteger class.

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



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


include/computedExpression
Integer.hpp
src/computedExpressionBoolean.cpp
src/computedExpressionFloat.cpp
src/computedExpressionInteger.cpp
src/program-execute.cpp
test/testGarbageCollected.cpp

Classes

- class Tang::ComputedExpressionInteger

  Represents an Integer that is the result of a computation.

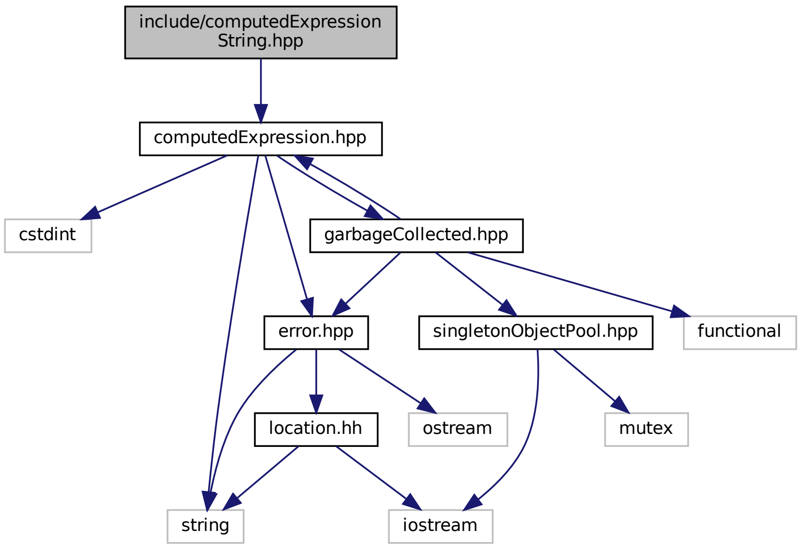
6.23.1 Detailed Description

Declare the Tang::ComputedExpressionInteger class.

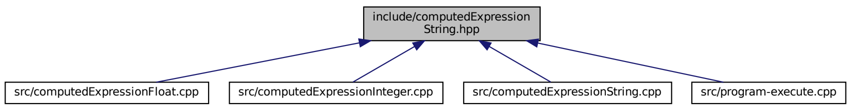
6.24 include/computedExpressionString.hpp File Reference

Declare the Tang::ComputedExpressionString class.

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



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


include/computedExpression String.hpp
src/computedExpressionFloat.cpp
src/computedExpressionInteger.cpp
src/computedExpressionString.cpp
src/program-execute.cpp

Classes

- class Tang::ComputedExpressionString

    Represents a String that is the result of a computation.

6.24.1 Detailed Description

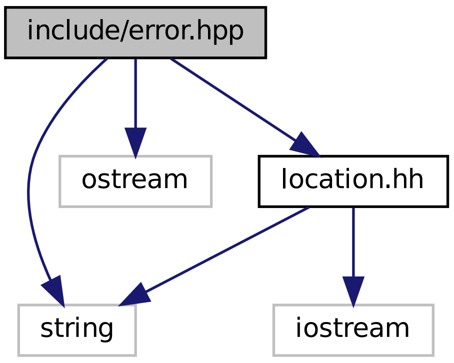
Declare the Tang::ComputedExpressionString class.

6.25 include/error.hpp File Reference

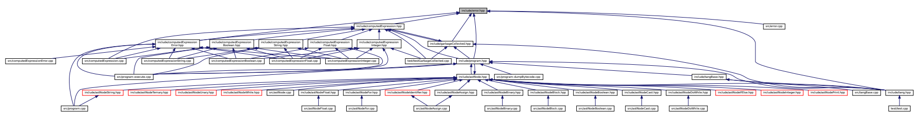
Declare the Tang::Error class used to describe syntax and runtime errors.

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

Include dependency graph for error.hpp:



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



Classes

- class Tang::Error

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

6.25.1 Detailed Description

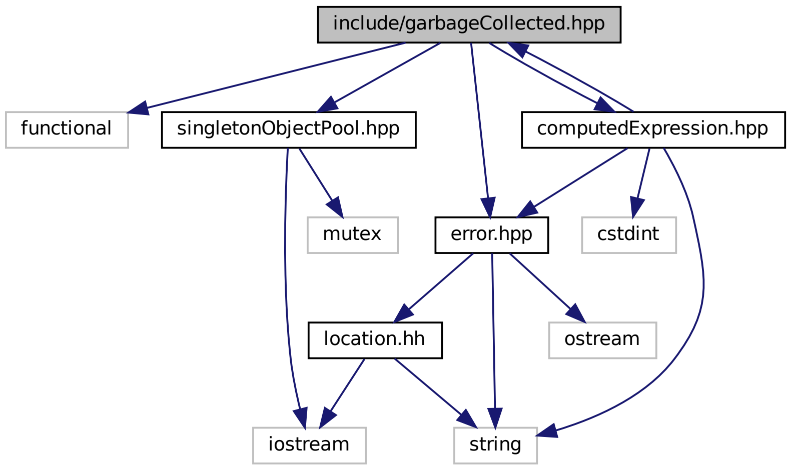
Declare the Tang::Error class used to describe syntax and runtime errors.

6.26 include/garbageCollected.hpp File Reference

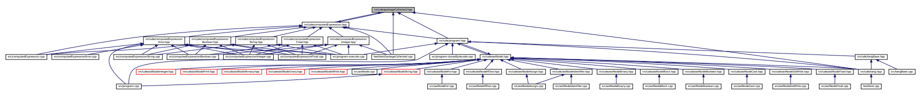
Declare the Tang::GarbageCollected class.

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

Include dependency graph for garbageCollected.hpp:



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



Classes

- class Tang::GarbageCollected

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

6.26.1 Detailed Description

Declare the Tang::GarbageCollected class.

6.27 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.

6.27.1 Detailed Description

Contains generic macros.

6.27.2 Macro Definition Documentation

6.27.2.1 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)) {}

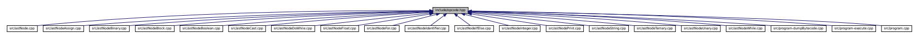
Parameters

x	The argument to be ignored.

6.28 include/opcode.hpp File Reference

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

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



Enumerations

- enum class Tang::Opcode {
  POP , PEEK , POKE , JMP ,
  JMPF , JMPF_POP , JMPT , JMPT_POP ,
  NULLVAL , INTEGER , FLOAT , BOOLEAN ,
  STRING , ADD , SUBTRACT , MULTIPLY ,
  DIVIDE , MODULO , NEGATIVE , NOT ,
  LT , LTE , GT , GTE ,
  EQ , NEQ , CASTINTEGER , CASTFLOAT ,
  CASTBOOLEAN , PRINT }

6.28.1 Detailed Description

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

6.28.2 Enumeration Type Documentation

6.28.2.1 Opcode

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

Enumerator

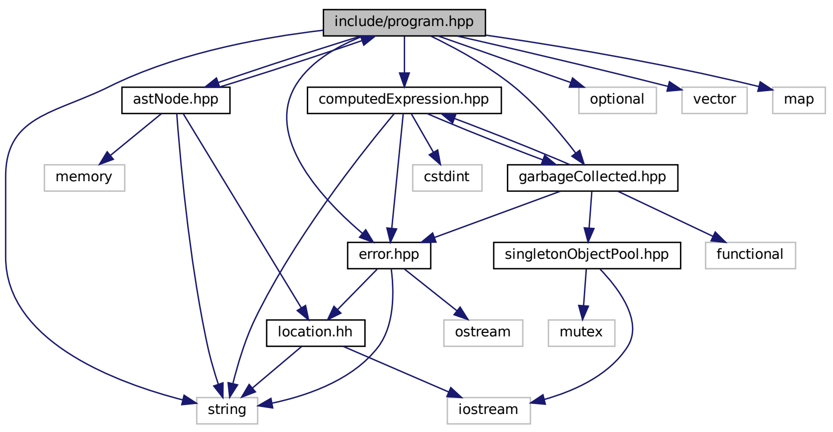
POP	Pop a val.
PEEK	Stack # (from fp): push val from stack #.
POKE	Stack # (from fp): Copy a val, store @ stack #.
JMP	PC #: set pc to PC #.
JMPF	PC #: read val, if false, set pc to PC #.
JMPF_POP	PC #: pop val, if false, set pc to PC #.
JMPT	PC #: read val, if true, set pc to PC #.
JMPT_POP	PC #: pop val, if true, set pc to PC #.
NULLVAL	Push a null onto the stack.
INTEGER	Push an integer onto the stack.
FLOAT	Push a floating point number onto the stack.
BOOLEAN	Push a boolean onto the stack.
STRING	Get len, char string: push string.
ADD	Pop rhs, pop lhs, push lhs + rhs.
SUBTRACT	Pop rhs, pop lhs, push lhs - rhs.
MULTIPLY	Pop rhs, pop lhs, push lhs $*$ rhs.
DIVIDE	Pop rhs, pop lhs, push lhs / rhs.
MODULO	Pop rhs, pop lhs, push lhs % rhs.
NEGATIVE	Pop val, push negative val.
NOT	Pop val, push logical not of val.
LT	Pop rhs, pop lhs, push lhs $<$ rhs.
LTE	Pop rhs, pop lhs, push lhs $<=$ rhs.
GT	Pop rhs, pop lhs, push lhs $>$ rhs.
GTE	Pop rhs, pop lhs, push lhs $>=$ rhs.
EQ	Pop rhs, pop lhs, push lhs == rhs.
NEQ	Pop rhs, pop lhs, push lhs != rhs.
CASTINTEGER	Pop a val, typecast to int, push.
CASTFLOAT	Pop a val, typecast to float, push.
CASTBOOLEAN	Pop a val, typecast to boolean, push.
PRINT	Pop val, print(val), push error or NULL.

6.29 include/program.hpp File Reference

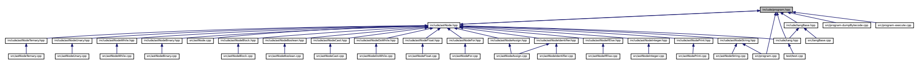
Declare the Tang::Program class used to compile and execute source code.

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

Include dependency graph for program.hpp:



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



Classes

- class Tang::Program

    Represents a compiled script or template that may be executed.

Typedefs

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

    Contains the Opcodes of a compiled program.

6.29.1 Detailed Description

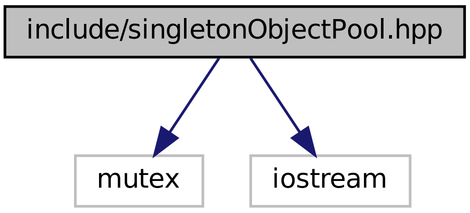
Declare the Tang::Program class used to compile and execute source code.

6.30 include/singletonObjectPool.hpp File Reference

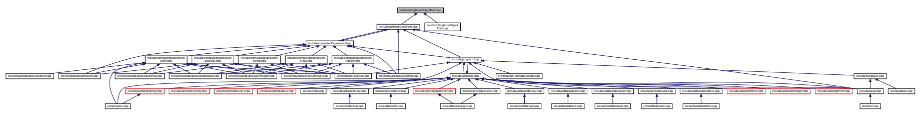
Declare the Tang::SingletonObjectPool class.

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

Include dependency graph for singletonObjectPool.hpp:



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



Classes

- class Tang::SingletonObjectPool< T >

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

Macros

- #define GROW 1024

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

6.30.1 Detailed Description

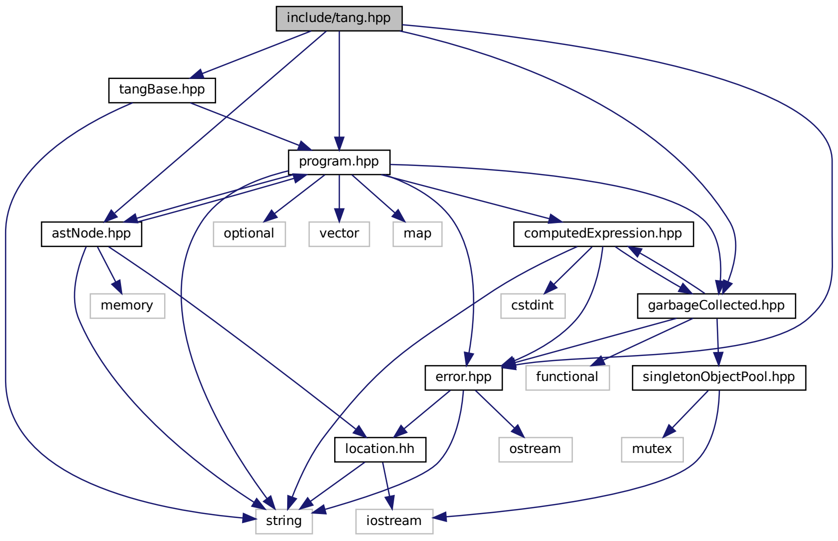
Declare the Tang::SingletonObjectPool class.

6.31 include/tang.hpp File Reference

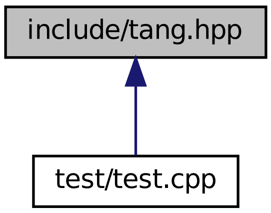
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

```
#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:



6.31.1 Detailed Description

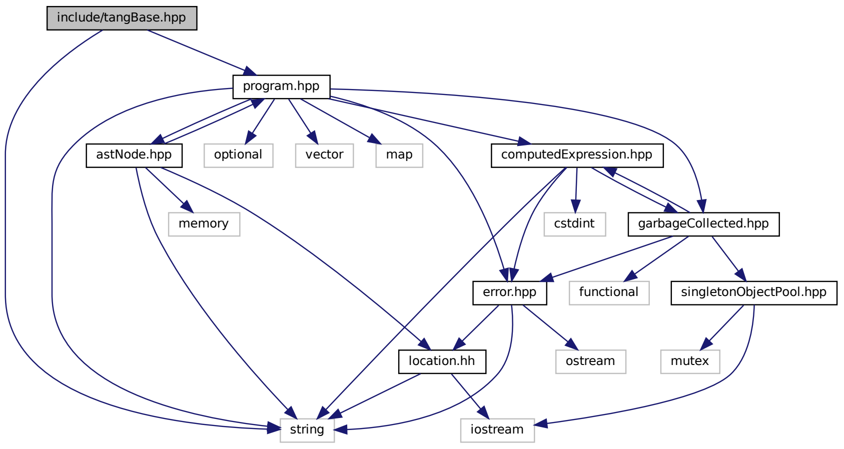
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

6.32 include/tangBase.hpp File Reference

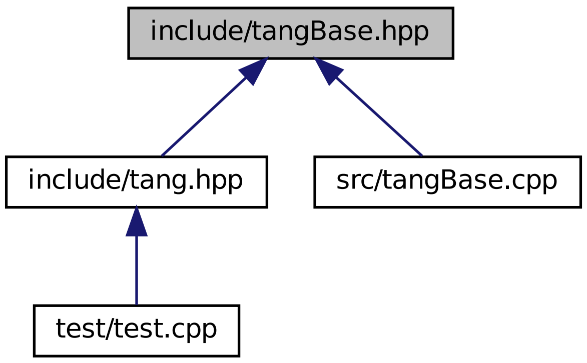
Declare the Tang::TangBase class used to interact with Tang.

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

Include dependency graph for tangBase.hpp:



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



Classes

- class Tang::TangBase

    The base class for the Tang programming language.

6.32.1 Detailed Description

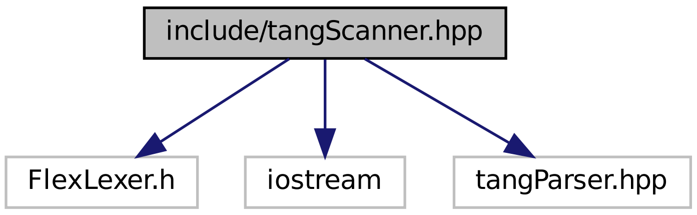
Declare the Tang::TangBase class used to interact with Tang.

6.33 include/tangScanner.hpp File Reference

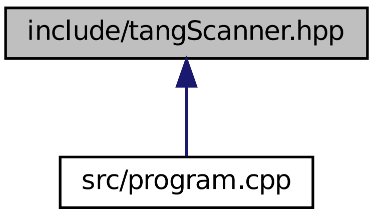
Declare the Tang::TangScanner used to tokenize a Tang script.

```
#include <FlexLexer.h>
#include <iostream>
#include "tangParser.hpp"
```

Include dependency graph for tangScanner.hpp:



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



Classes

- class Tang::TangScanner

  The Flex lexer class for the main Tang language.

Macros

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

6.33.1 Detailed Description

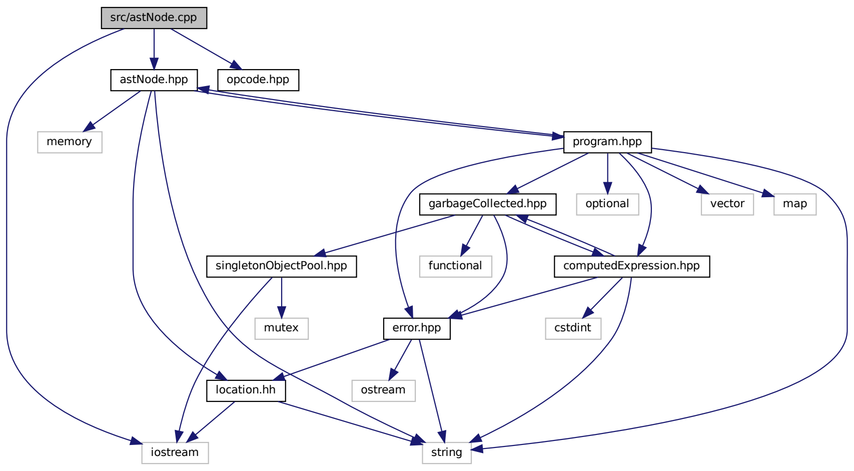
Declare the Tang::TangScanner used to tokenize a Tang script.

6.34 src/astNode.cpp File Reference

Define the Tang::AstNode class.

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

Include dependency graph for astNode.cpp:



6.34.1 Detailed Description

Define the Tang::AstNode class.

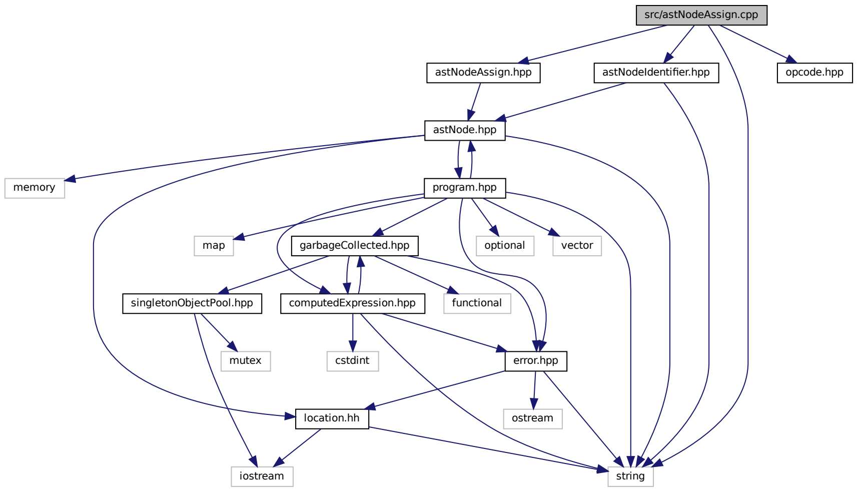
6.35 src/astNodeAssign.cpp File Reference

Define the Tang::AstNodeAssign class.

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

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

Include dependency graph for astNodeAssign.cpp:



6.35.1 Detailed Description

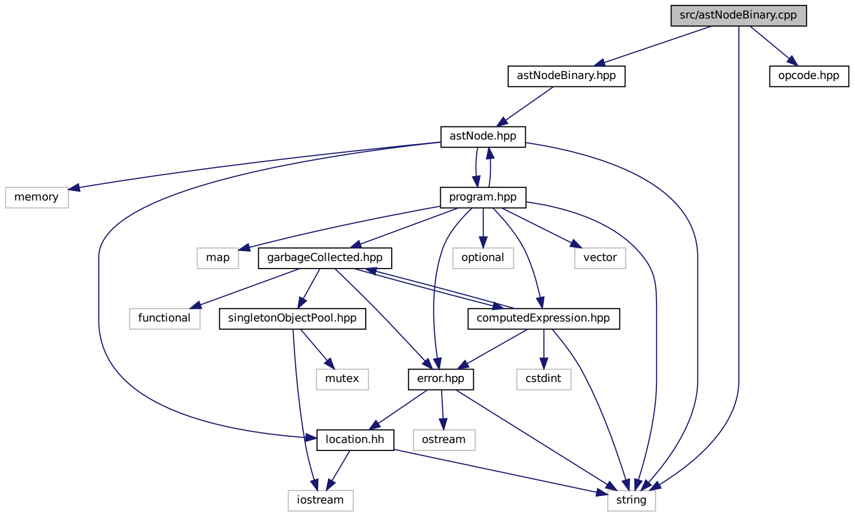
Define the Tang::AstNodeAssign class.

6.36 src/astNodeBinary.cpp File Reference

Define the Tang::AstNodeBinary class.

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

Include dependency graph for astNodeBinary.cpp:



6.36.1 Detailed Description

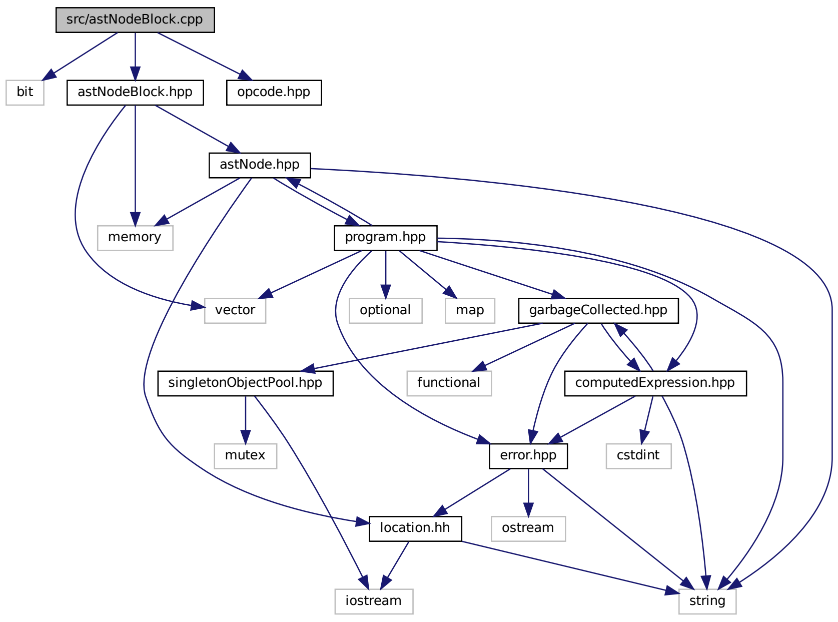
Define the Tang::AstNodeBinary class.

6.37 src/astNodeBlock.cpp File Reference

Define the Tang::AstNodeBlock class.

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

Include dependency graph for astNodeBlock.cpp:



6.37.1 Detailed Description

Define the Tang::AstNodeBlock class.

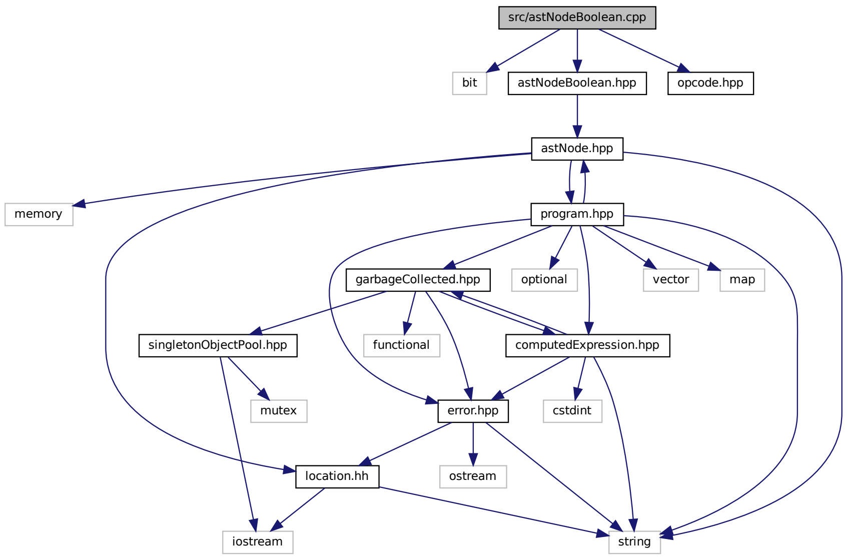
6.38 src/astNodeBoolean.cpp File Reference

Define the Tang::AstNodeBoolean class.

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

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

Include dependency graph for astNodeBoolean.cpp:



6.38.1 Detailed Description

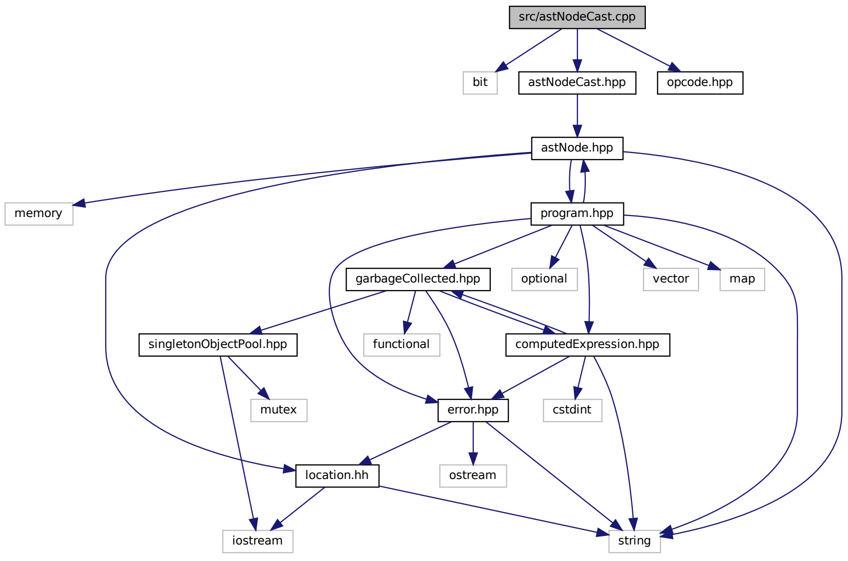
Define the Tang::AstNodeBoolean class.

6.39 src/astNodeCast.cpp File Reference

Define the Tang::AstNodeCast class.

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

Include dependency graph for astNodeCast.cpp:



6.39.1 Detailed Description

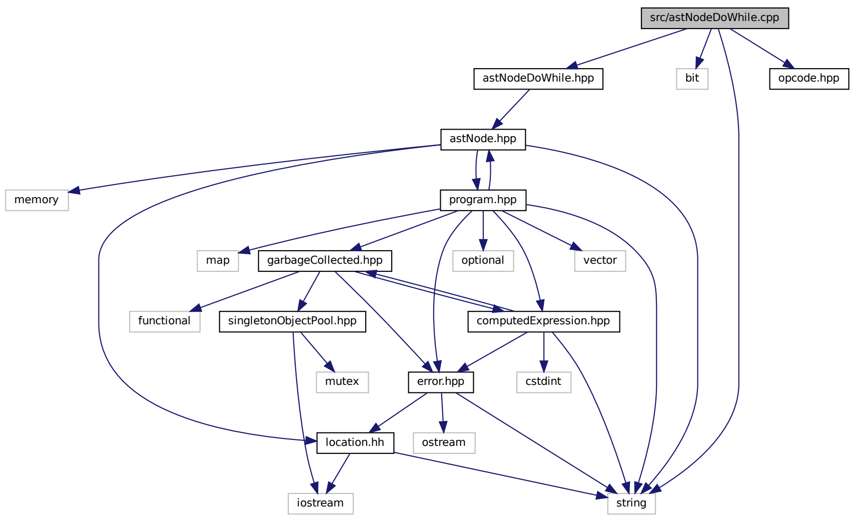
Define the Tang::AstNodeCast class.

6.40 src/astNodeDoWhile.cpp File Reference

Define the Tang::AstNodeDoWhile class.

```
#include <string>
#include <bit>
#include "astNodeDoWhile.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeDoWhile.cpp:



6.40.1 Detailed Description

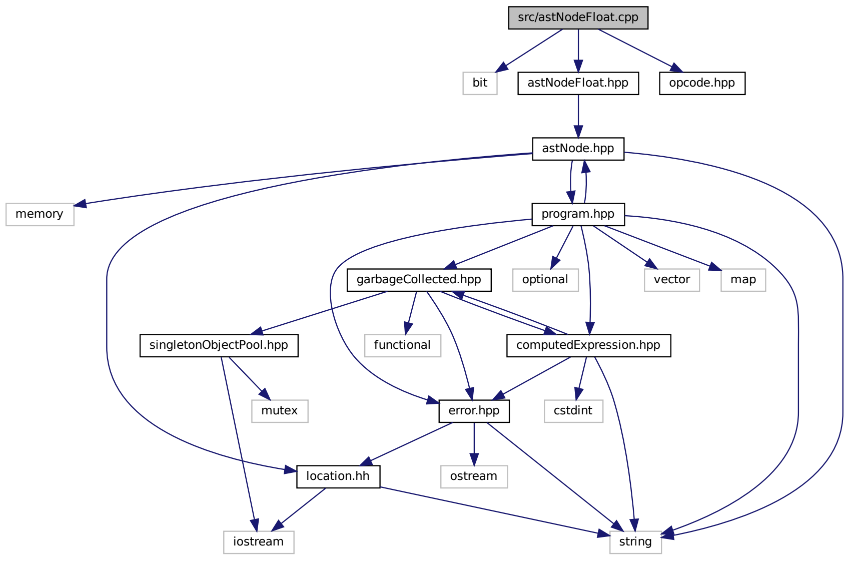
Define the Tang::AstNodeDoWhile class.

6.41 src/astNodeFloat.cpp File Reference

Define the Tang::AstNodeFloat class.

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

Include dependency graph for astNodeFloat.cpp:



6.41.1 Detailed Description

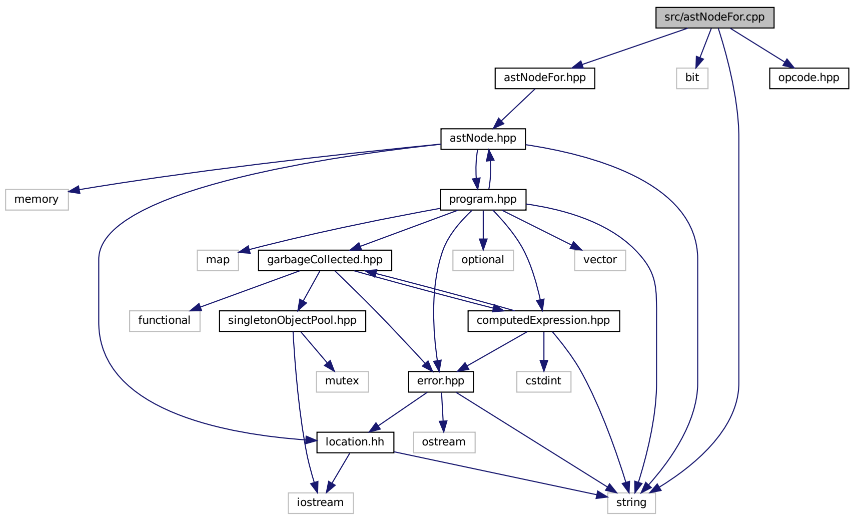
Define the Tang::AstNodeFloat class.

6.42 src/astNodeFor.cpp File Reference

Define the Tang::AstNodeFor class.

```
#include <string>
#include <bit>
#include "astNodeFor.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeFor.cpp:



6.42.1 Detailed Description

Define the Tang::AstNodeFor class.

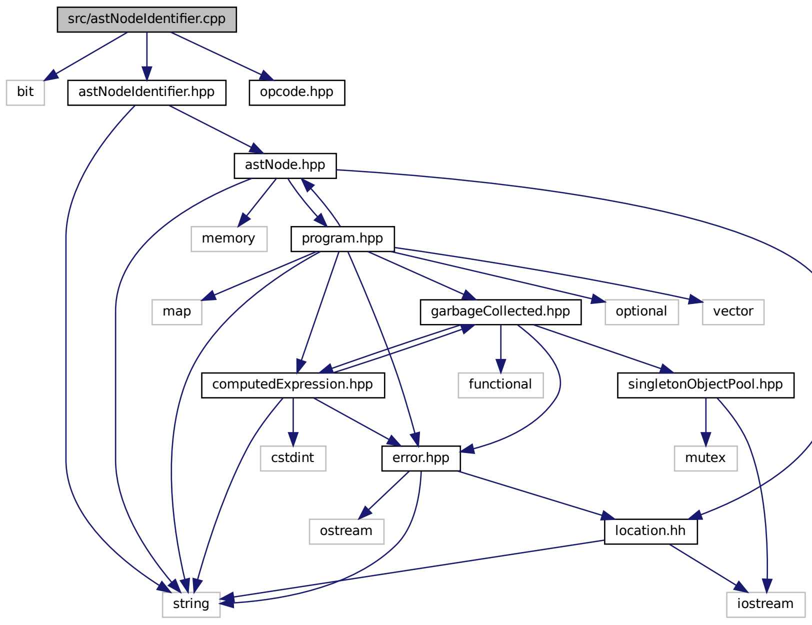
6.43 src/astNodeIdentifier.cpp File Reference

Define the Tang::AstNodeIdentifier class.

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

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

Include dependency graph for astNodeIdentifier.cpp:



6.43.1 Detailed Description

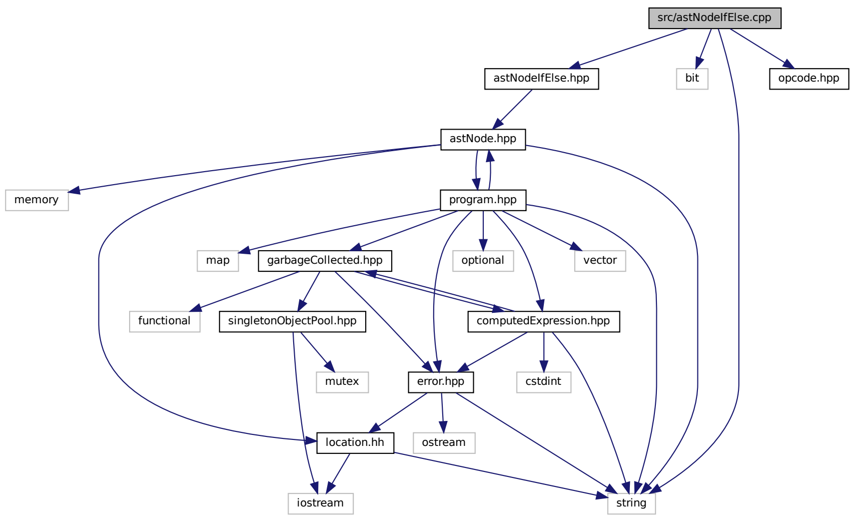
Define the Tang::AstNodeIdentifier class.

6.44 src/astNodeIfElse.cpp File Reference

Define the Tang::AstNodeIfElse class.

```
#include <string>
#include <bit>
#include "astNodeIfElse.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeIfElse.cpp:



6.44.1 Detailed Description

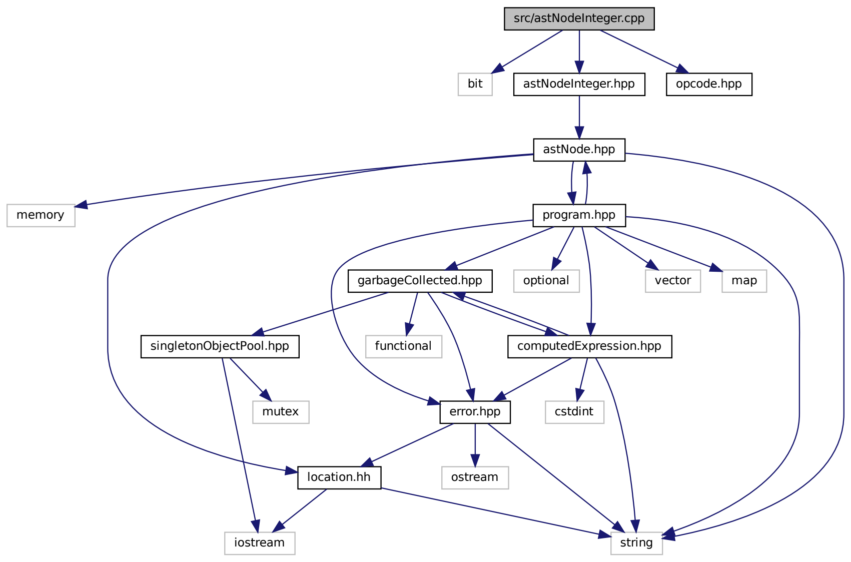
Define the Tang::AstNodeIfElse class.

6.45 src/astNodeInteger.cpp File Reference

Define the Tang::AstNodeInteger class.

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

Include dependency graph for astNodeInteger.cpp:



6.45.1 Detailed Description

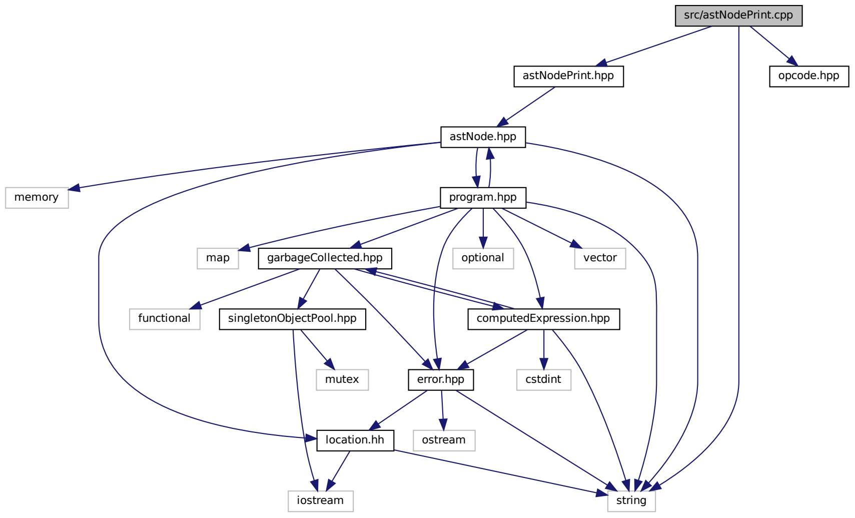
Define the Tang::AstNodeInteger class.

6.46 src/astNodePrint.cpp File Reference

Define the Tang::AstNodePrint class.

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

Include dependency graph for astNodePrint.cpp:



6.46.1 Detailed Description

Define the Tang::AstNodePrint class.

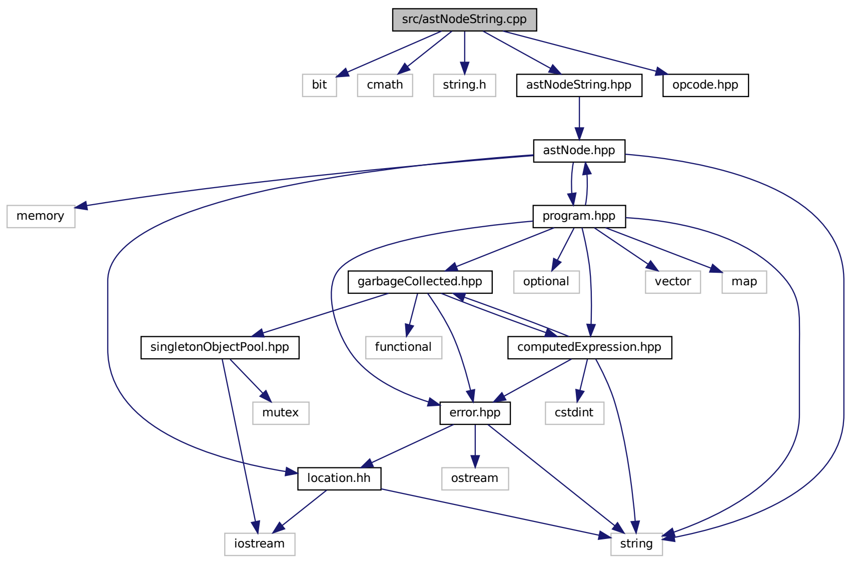
6.47 src/astNodeString.cpp File Reference

Define the Tang::AstNodeString class.

```
#include <bit>
#include <cmath>
#include <string.h>
#include "astNodeString.hpp"
```

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

Include dependency graph for astNodeString.cpp:



6.47.1 Detailed Description

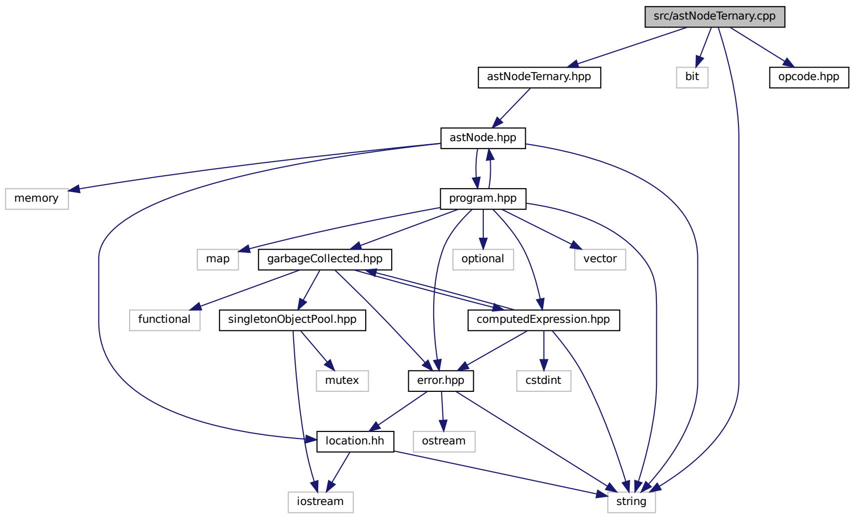
Define the Tang::AstNodeString class.

6.48 src/astNodeTernary.cpp File Reference

Define the Tang::AstNodeTernary class.

```
#include <string>
#include <bit>
#include "astNodeTernary.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeTernary.cpp:



6.48.1 Detailed Description

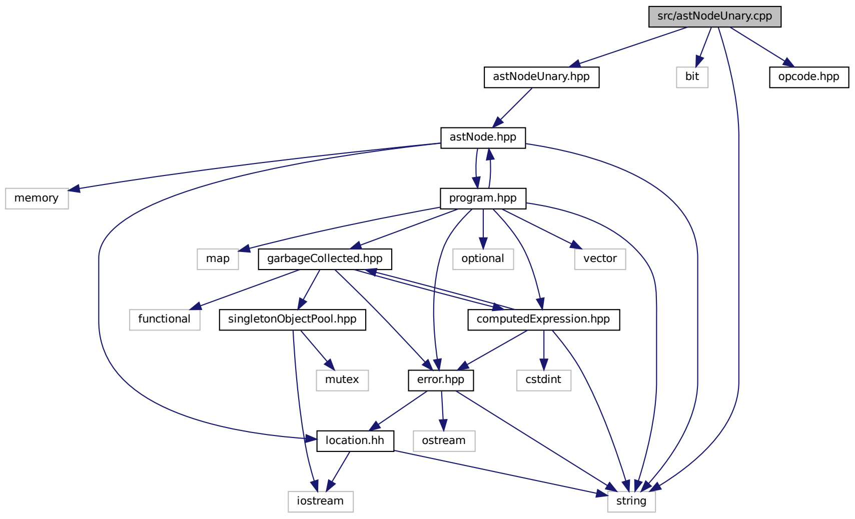
Define the Tang::AstNodeTernary class.

6.49 src/astNodeUnary.cpp File Reference

Define the Tang::AstNodeUnary class.

```
#include <string>
#include <bit>
#include "astNodeUnary.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeUnary.cpp:



6.49.1 Detailed Description

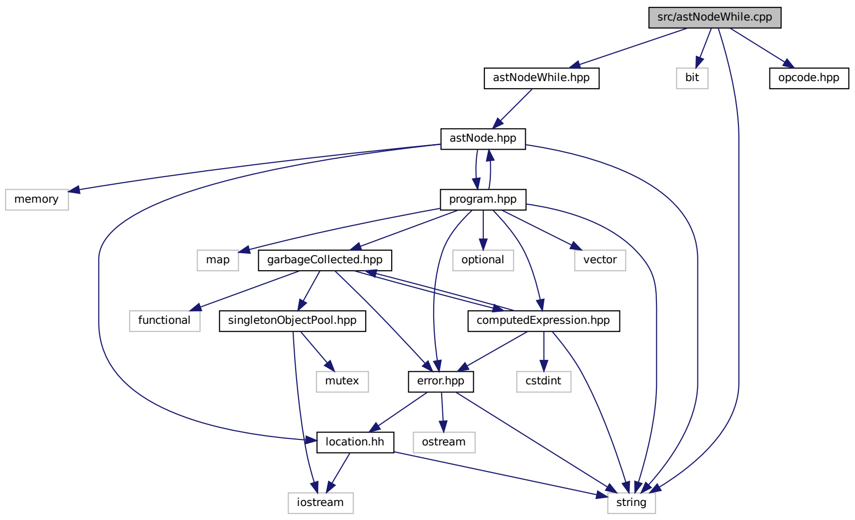
Define the Tang::AstNodeUnary class.

6.50 src/astNodeWhile.cpp File Reference

Define the Tang::AstNodeWhile class.

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

Include dependency graph for astNodeWhile.cpp:



6.50.1 Detailed Description

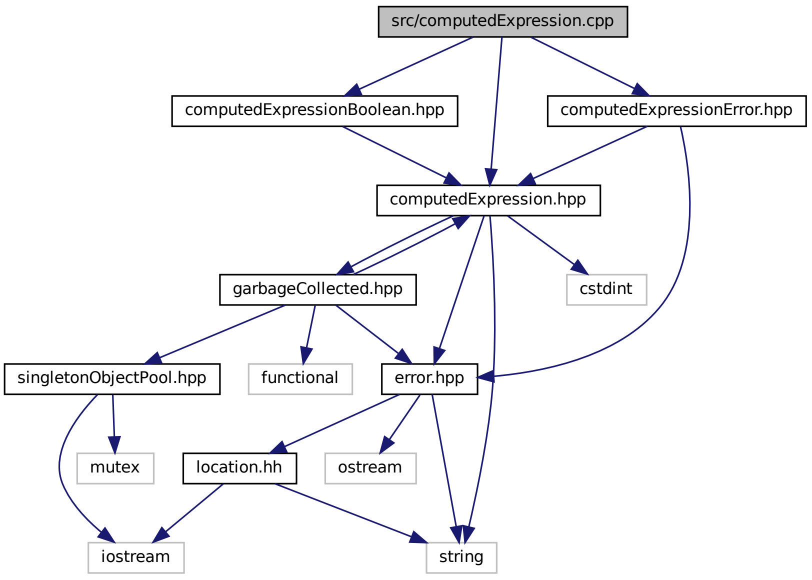
Define the Tang::AstNodeWhile class.

6.51 src/computedExpression.cpp File Reference

Define the Tang::ComputedExpression class.

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

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



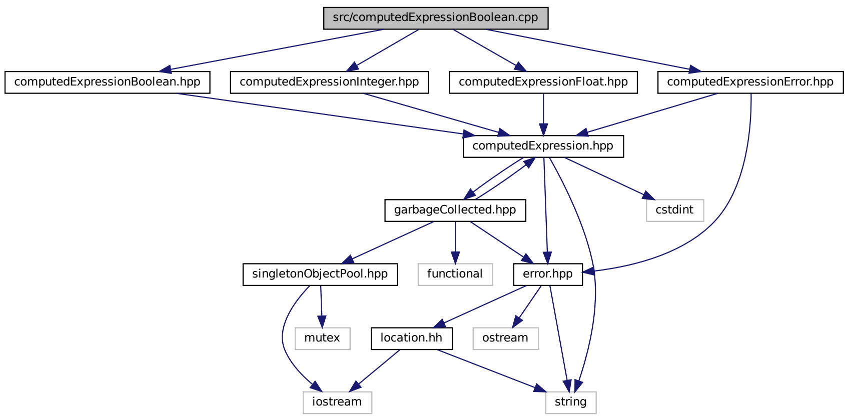
6.51.1 Detailed Description

Define the Tang::ComputedExpression class.

6.52 src/computedExpressionBoolean.cpp File Reference

Define the Tang::ComputedExpressionBoolean class.

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


src/computedExpressionBoolean.cpp
computedExpressionBoolean.hpp
computedExpressionInteger.hpp
computedExpressionFloat.hpp
computedExpressionError.hpp
computedExpression.hpp
garbageCollected.hpp
cstdint
singletonObjectPool.hpp
functional
error.hpp
mutex
location.hh
ostream
iostream
string

6.52.1 Detailed Description

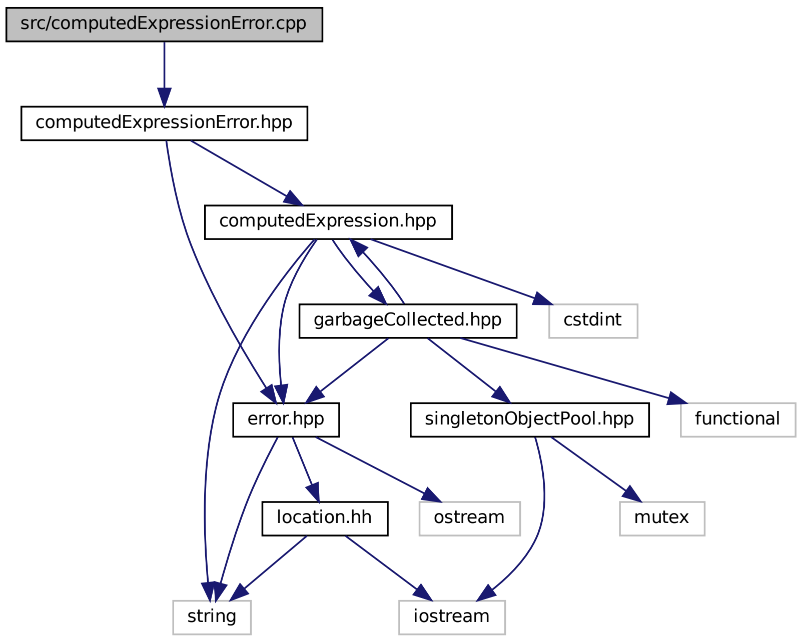
Define the Tang::ComputedExpressionBoolean class.

6.53 src/computedExpressionError.cpp File Reference

Define the Tang::ComputedExpressionError class.

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

Include dependency graph for computedExpressionError.cpp:



6.53.1 Detailed Description

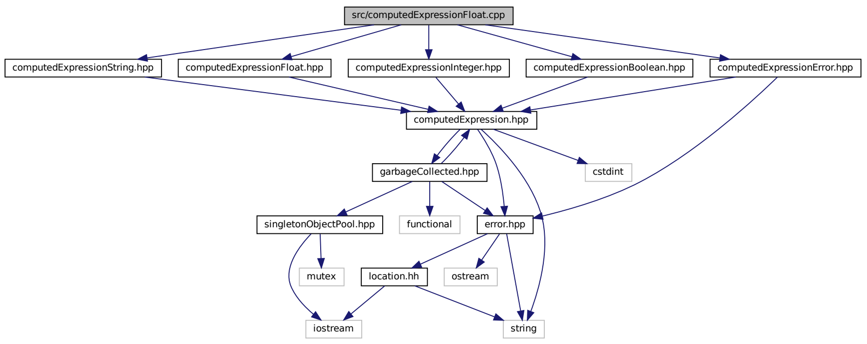
Define the Tang::ComputedExpressionError class.

6.54 src/computedExpressionFloat.cpp File Reference

Define the Tang::ComputedExpressionFloat class.

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

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


src/computedExpressionFloat.cpp
computedExpressionString.hpp
computedExpressionFloat.hpp
computedExpressionInteger.hpp
computedExpressionBoolean.hpp
computedExpressionError.hpp
computedExpression.hpp
garbageCollected.hpp
cstdint
singletonObjectPool.hpp
functional
error.hpp
mutex
location.hh
ostream
iostream
string

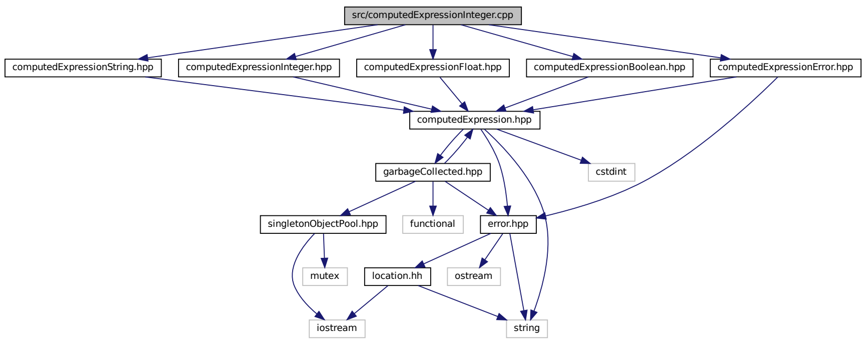
6.54.1 Detailed Description

Define the Tang::ComputedExpressionFloat class.

6.55 src/computedExpressionInteger.cpp File Reference

Define the Tang::ComputedExpressionInteger class.

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


src/computedExpressionInteger.cpp
computedExpressionString.hpp
computedExpressionInteger.hpp
computedExpressionFloat.hpp
computedExpressionBoolean.hpp
computedExpressionError.hpp
computedExpression.hpp
garbageCollected.hpp
cstdint
singletonObjectPool.hpp
functional
error.hpp
mutex
location.hh
ostream
iostream
string

6.55.1 Detailed Description

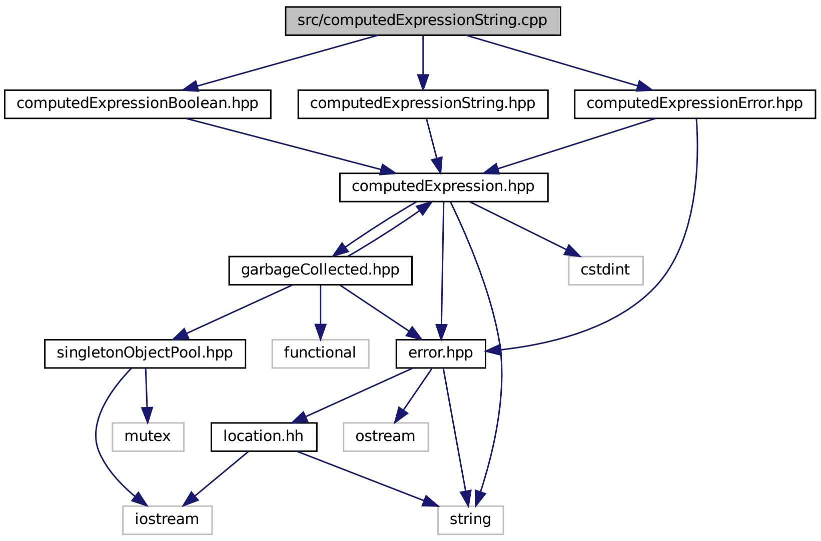
Define the Tang::ComputedExpressionInteger class.

6.56 src/computedExpressionString.cpp File Reference

Define the Tang::ComputedExpressionString class.

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

Include dependency graph for computedExpressionString.cpp:



6.56.1 Detailed Description

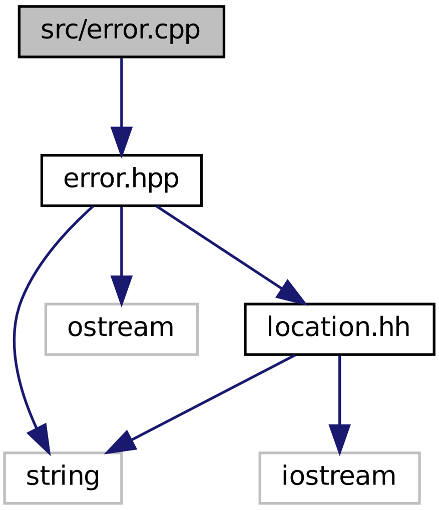
Define the Tang::ComputedExpressionString class.

6.57 src/error.cpp File Reference

Define the Tang::Error class.

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

Include dependency graph for error.cpp:



Functions

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

6.57.1 Detailed Description

Define the Tang::Error class.

6.57.2 Function Documentation

6.57.2.1 operator<<()

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

Parameters

out	The output stream.
error	The Error object.

Returns

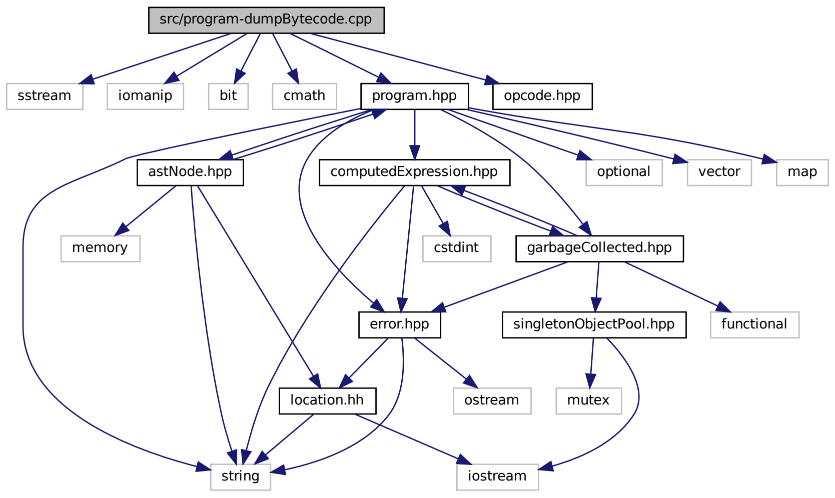
The output stream.

6.58 src/program-dumpBytecode.cpp File Reference

Define the Tang::Program::dumpBytecode method.

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

Include dependency graph for program-dumpBytecode.cpp:



Macros

- #define DUMPPROGRAMCHECK(x)

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

6.58.1 Detailed Description

Define the Tang::Program::dumpBytecode method.

6.58.2 Macro Definition Documentation

6.58.2.1 DUMPPROGRAMCHECK

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

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

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

If the vector is not large enough, an error message is appended to the output string and no further opcodes are printed.

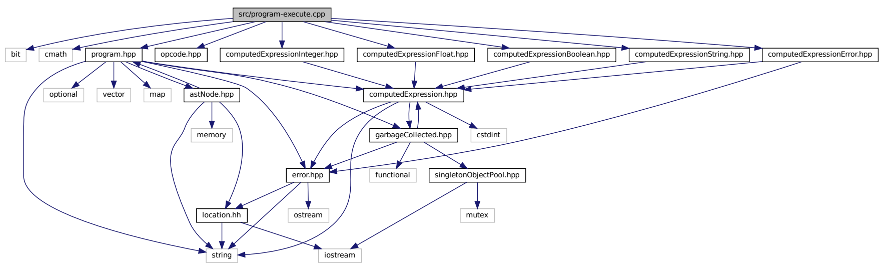
Parameters

x	The number of additional vector entries that should exist.

6.59 src/program-execute.cpp File Reference

Define the Tang::Program::execute method.

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

6.59.1 Detailed Description

Define the Tang::Program::execute method.

6.59.2 Macro Definition Documentation

6.59.2.1 EXECUTEPROGRAMCHECK

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

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

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

Parameters

x	The number of additional vector entries that should exist.

6.59.2.2 STACKCHECK

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

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

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

Parameters

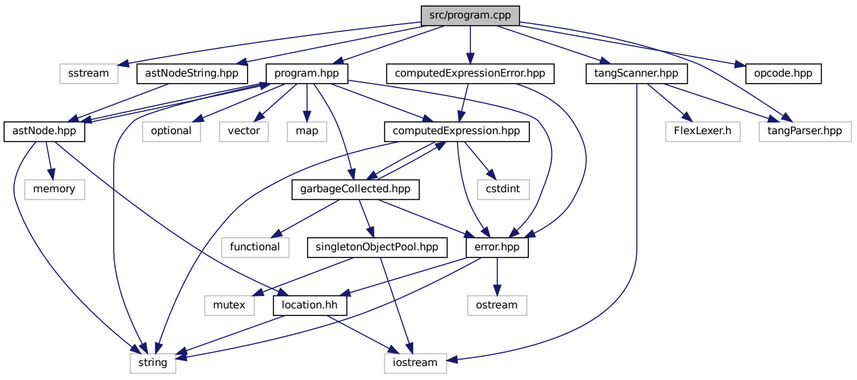
x	The number of entries that should exist in the stack.

6.60 src/program.cpp File Reference

Define the Tang::Program class.

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

Include dependency graph for program.cpp:



6.60.1 Detailed Description

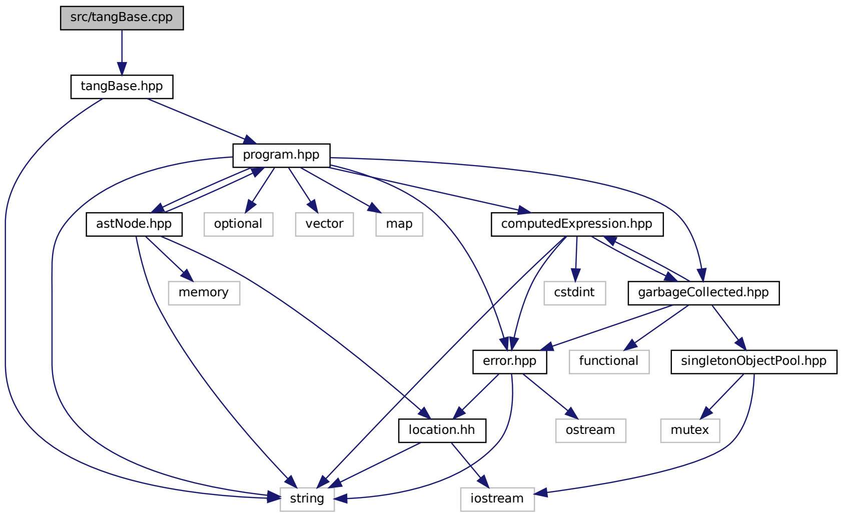
Define the Tang::Program class.

6.61 src/tangBase.cpp File Reference

Define the Tang::TangBase class.

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

Include dependency graph for tangBase.cpp:



6.61.1 Detailed Description

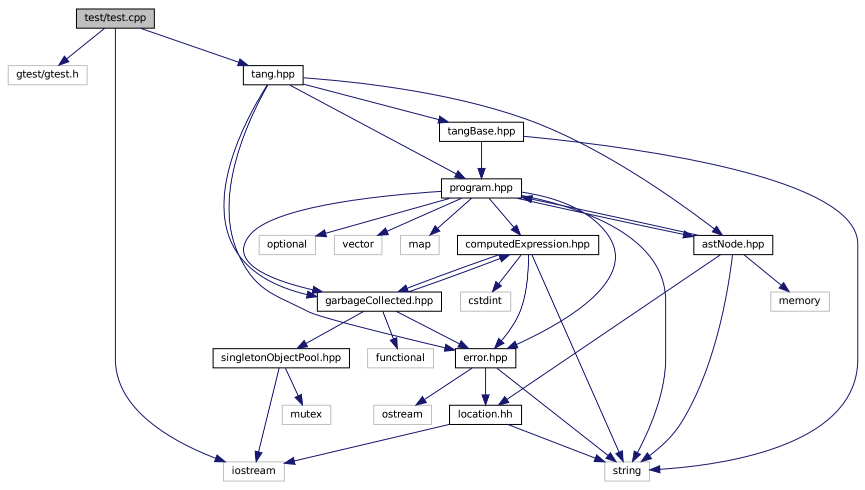
Define the Tang::TangBase class.

6.62 test/test.cpp File Reference

Test the general language behaviors.

```
#include <gtest/gtest.h>
#include <iostream>
#include "tang.hpp"
```

Include dependency graph for test.cpp:



Functions

- **TEST** (Declare, Null)
- **TEST** (Declare, Integer)
- **TEST** (Declare, Float)
- **TEST** (Declare, Boolean)
- **TEST** (Declare, String)
- **TEST** (Expression, Add)
- **TEST** (Expression, Subtract)
- **TEST** (Expression, Multiplication)
- **TEST** (Expression, Division)
- **TEST** (Expression, Modulo)
- **TEST** (Expression, UnaryMinus)
- **TEST** (Expression, Parentheses)
- **TEST** (Expression, TypeCast)
- **TEST** (Expression, Not)
- **TEST** (Expression, LessThan)
- **TEST** (Expression, LessThanEqual)
- **TEST** (Expression, GreaterThan)
- **TEST** (Expression, GreaterThanEqual)

- **TEST** (Expression, Equal)
- **TEST** (Expression, NotEqual)
- **TEST** (Expression, And)
- **TEST** (Expression, Or)
- **TEST** (Expression, Ternary)
- **TEST** (CodeBlock, Statements)
- **TEST** (Assign, Identifier)
- **TEST** (ControlFlow, IfElse)
- **TEST** (ControlFlow, While)
- **TEST** (ControlFlow, DoWhile)
- **TEST** (ControlFlow, For)
- **TEST** (Print, Default)
- int **main** (int argc, char ∗∗argv)

6.62.1 Detailed Description

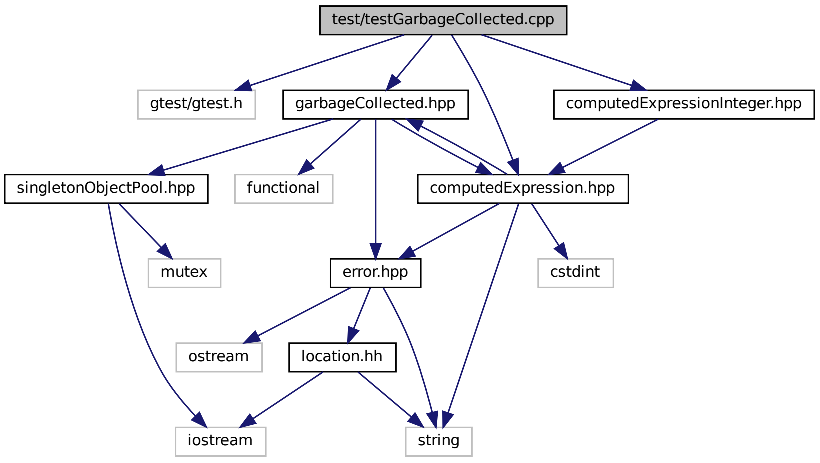
Test the general language behaviors.

6.63 test/testGarbageCollected.cpp File Reference

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

```
#include <gtest/gtest.h>
#include "garbageCollected.hpp"
#include "computedExpression.hpp"
#include "computedExpressionInteger.hpp"
```

Include dependency graph for testGarbageCollected.cpp:



Functions

- **TEST** (Create, Access)
- **TEST** (RuleOfFive, CopyConstructor)
- **TEST** (Recycle, ObjectIsRecycled)
- **TEST** (Recycle, ObjectIsNotRecycled)
- int **main** (int argc, char ∗∗argv)

6.63.1 Detailed Description

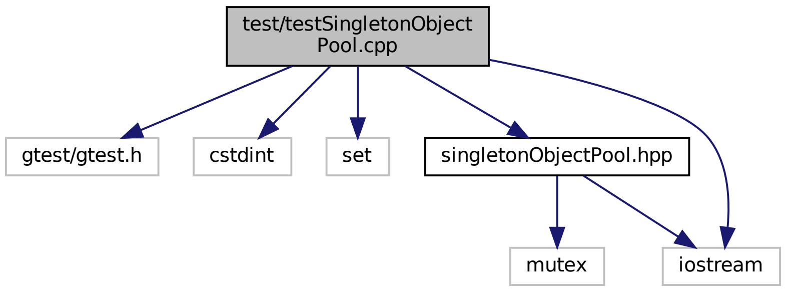
Test the generic behavior of the Tang::GarbageCollected class.

6.64 test/testSingletonObjectPool.cpp File Reference

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

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

Include dependency graph for testSingletonObjectPool.cpp:



Functions

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

6.64.1 Detailed Description

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

Index