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

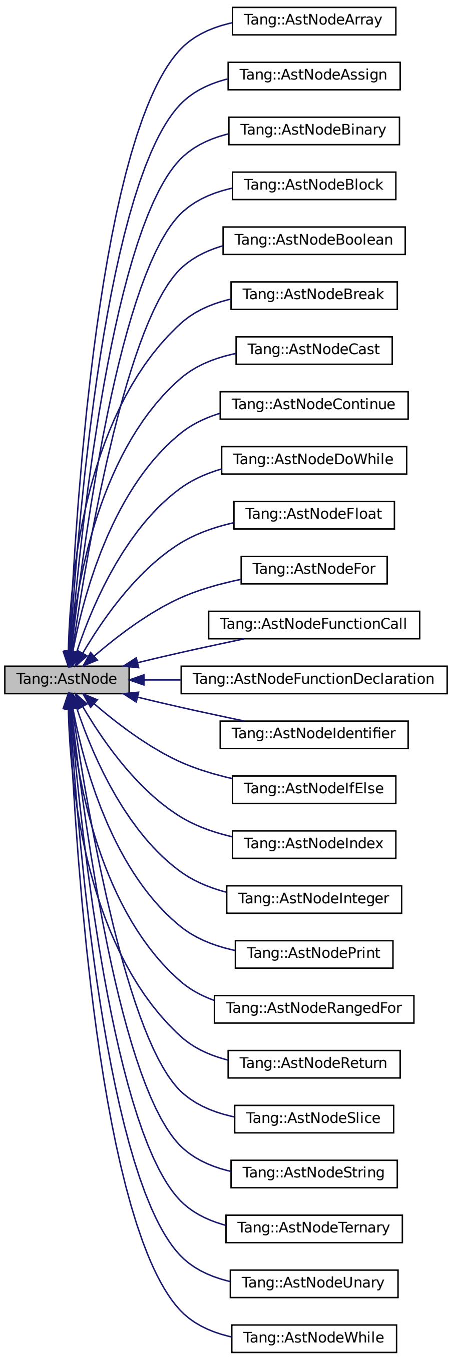
Class Documentation

5.1 Tang::AstNode Class Reference

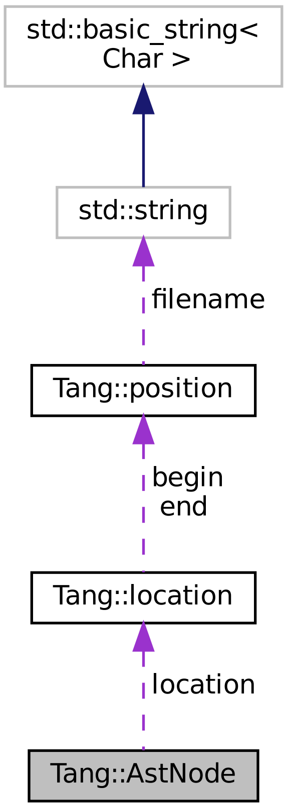
Base class for representing nodes of an Abstract Syntax Tree (AST).

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

Inheritance diagram for Tang::AstNode:



Collaboration diagram for Tang::AstNode:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const

  Run any preprocess analysis needed before compilation.

Private Attributes

- Tang::location location

    The location associated with this node.

5.1.1 Detailed Description

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

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 Member Enumeration Documentation

5.1.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.1.3 Constructor & Destructor Documentation

5.1.3.1 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.

5.1.4 Member Function Documentation

5.1.4.1 compile()

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

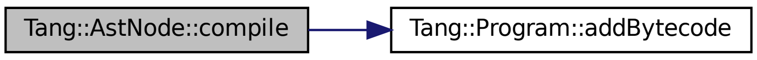
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeSlice, Tang::AstNodeReturn, Tang::AstNodeRangedFor, Tang::AstNodePrint, Tang::AstNodeInteger, Tang::AstNodeIndex, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeFloat, Tang::AstNodeDoWhile, Tang::AstNodeContinue, Tang::AstNodeCast, Tang::AstNodeBreak, Tang::AstNodeBoolean, Tang::AstNodeBlock, Tang::AstNodeBinary, Tang::AstNodeAssign, and Tang::AstNodeArray.

Here is the call graph for this function:



5.1.4.2 compilePreprocess()

```
void AstNode::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeSlice, Tang::AstNodeReturn, Tang::AstNodeRangedFor, Tang::AstNodePrint, Tang::AstNodeIndex, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBlock, Tang::AstNodeBinary, Tang::AstNodeAssign, and Tang::AstNodeArray.

5.1.4.3 dump()

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

Return a string that describes the contents of the node.

Parameters

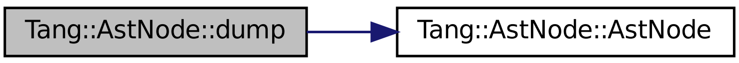
indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeSlice, Tang::AstNodeReturn, Tang::AstNodeRangedFor, Tang::AstNodePrint, Tang::AstNodeInteger, Tang::AstNodeIndex, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeFloat, Tang::AstNodeDoWhile, Tang::AstNodeContinue, Tang::AstNodeCast, Tang::AstNodeBreak, Tang::AstNodeBoolean, Tang::AstNodeBlock, Tang::AstNodeBinary, Tang::AstNodeAssign, and Tang::AstNodeArray.

Here is the call graph for this function:



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

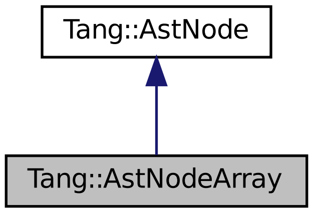
- include/astNode.hpp
- src/astNode.cpp

5.2 Tang::AstNodeArray Class Reference

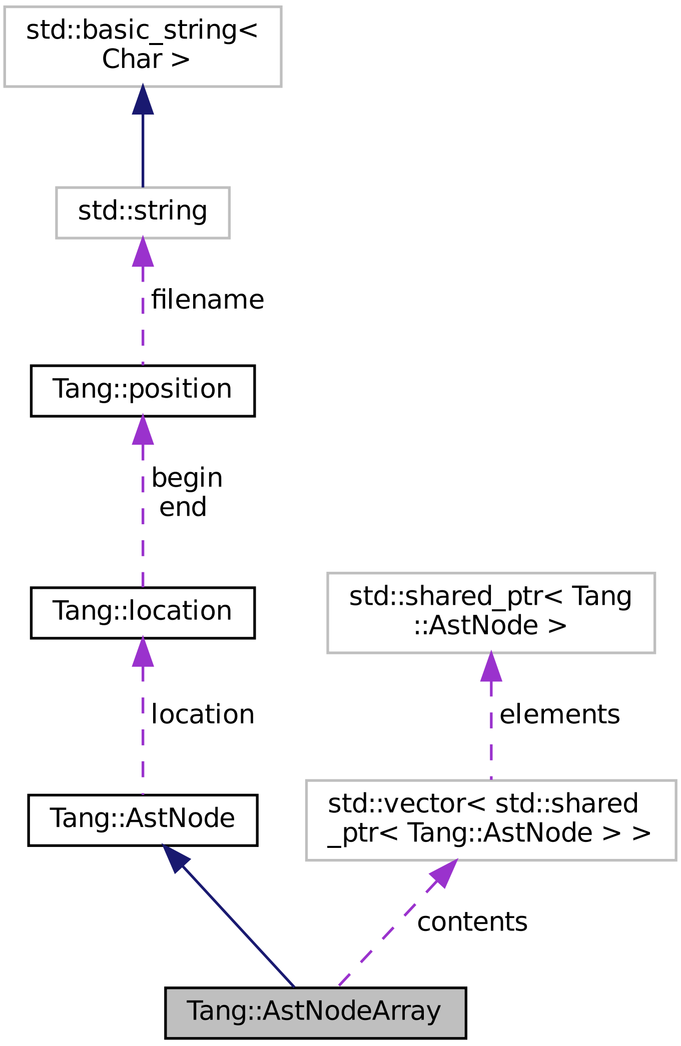
An AstNode that represents an array literal.

```
#include <astNodeArray.hpp>
```

Inheritance diagram for Tang::AstNodeArray:



Collaboration diagram for Tang::AstNodeArray:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeArray (std::vector< std::shared_ptr< Tang::AstNode >> contents, 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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- std::vector< std::shared_ptr< Tang::AstNode > > contents

  The contents of the array.
- Tang::location location

  The location associated with this node.

5.2.1 Detailed Description

An AstNode that represents an array literal.

5.2.2 Member Enumeration Documentation

5.2.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.2.3 Constructor & Destructor Documentation

5.2.3.1 AstNodeArray()

```
AstNodeArray::AstNodeArray (
             std::vector< std::shared_ptr< Tang::AstNode >> contents,
             Tang::location location )
```

The constructor.

Parameters

contents	The contents of the array.
location	The location associated with the expression.

5.2.4 Member Function Documentation

5.2.4.1 compile()

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

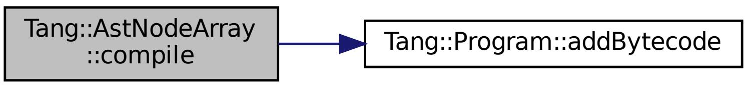
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.2.4.2 compilePreprocess()

```
void AstNodeArray::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.2.4.3 dump()

```
string AstNodeArray::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:

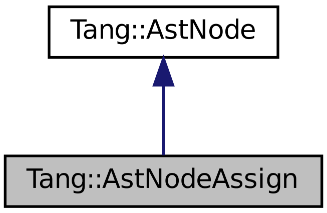
- include/astNodeArray.hpp
- src/astNodeArray.cpp

5.3 Tang::AstNodeAssign Class Reference

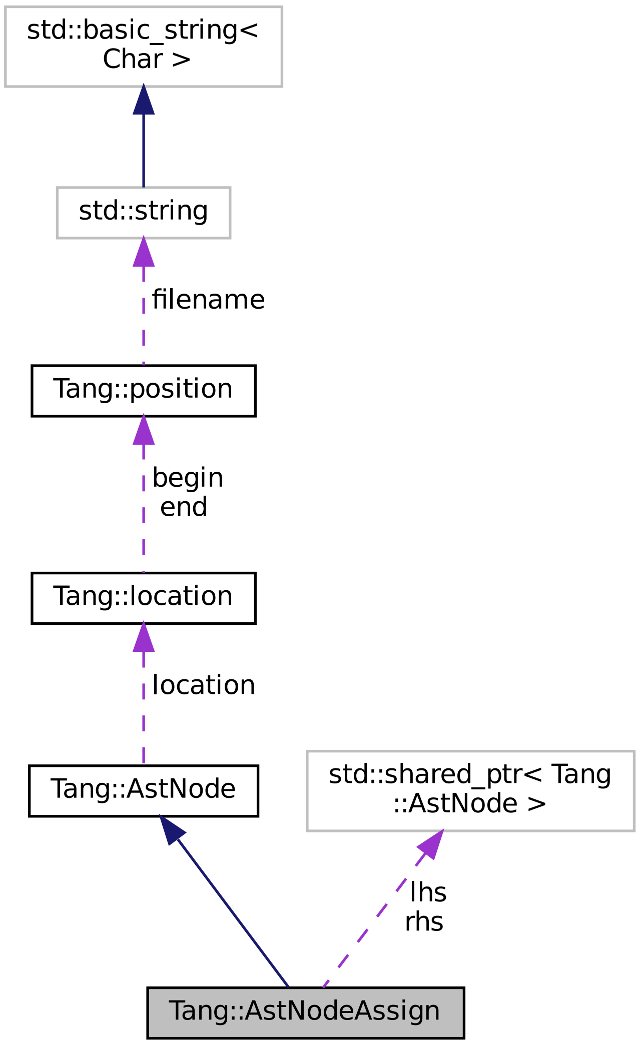
An AstNode that represents a binary expression.

```
#include <astNodeAssign.hpp>
```

Inheritance diagram for Tang::AstNodeAssign:



Collaboration diagram for Tang::AstNodeAssign:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- std::shared_ptr< AstNode > lhs

  The left hand side expression.
- std::shared_ptr< AstNode > rhs

  The right hand side expression.
- Tang::location location

  The location associated with this node.

5.3.1 Detailed Description

An AstNode that represents a binary expression.

5.3.2 Member Enumeration Documentation

5.3.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.3.3 Constructor & Destructor Documentation

5.3.3.1 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.

5.3.4 Member Function Documentation

5.3.4.1 compile()

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

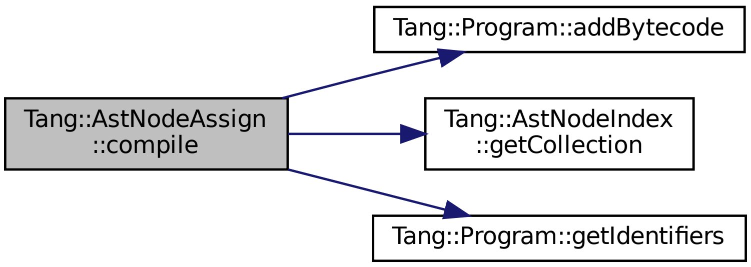
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::Program::addBytecode
Tang::AstNodeAssign ::compile
Tang::AstNodeIndex ::getCollection
Tang::Program::getIdentifiers

5.3.4.2 compilePreprocess()

```
void AstNodeAssign::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.3.4.3 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:

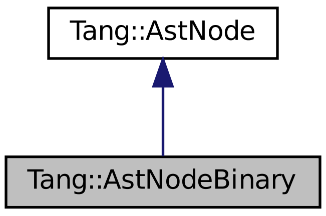
- include/astNodeAssign.hpp
- src/astNodeAssign.cpp

5.4 Tang::AstNodeBinary Class Reference

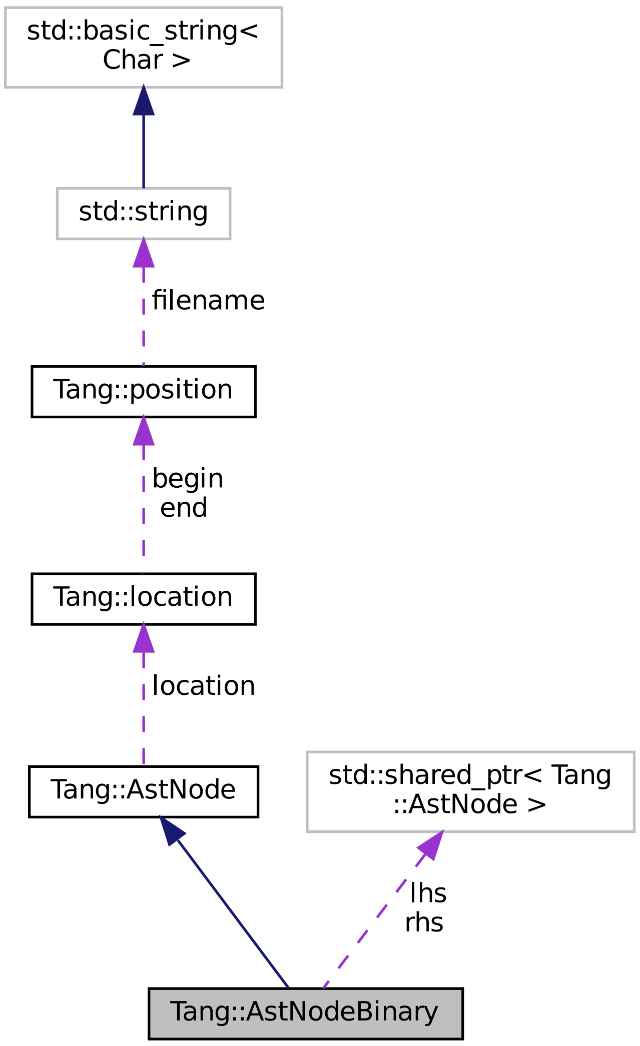
An AstNode that represents a binary expression.

`#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.
- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- Operation op

  The binary operation performed.
- std::shared_ptr< AstNode > lhs

  The left hand side expression.
- std::shared_ptr< AstNode > rhs

  The right hand side expression.
- Tang::location location

  The location associated with this node.

5.4.1 Detailed Description

An AstNode that represents a binary expression.

5.4.2 Member Enumeration Documentation

5.4.2.1 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.

5.4.2.2 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.4.3 Constructor & Destructor Documentation

5.4.3.1 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.

5.4.4 Member Function Documentation

5.4.4.1 compile()

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

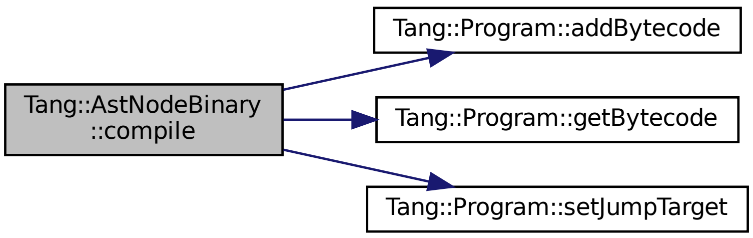
Compile the ast of the provided Tang::Program.

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

5.4.4.2 compilePreprocess()

```
void AstNodeBinary::compilePreprocess (
             Program & program,
             PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.4.4.3 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:

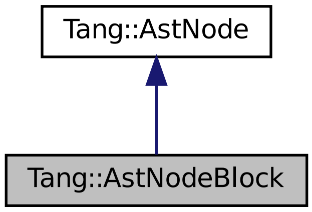
- include/astNodeBinary.hpp
- src/astNodeBinary.cpp

5.5 Tang::AstNodeBlock Class Reference

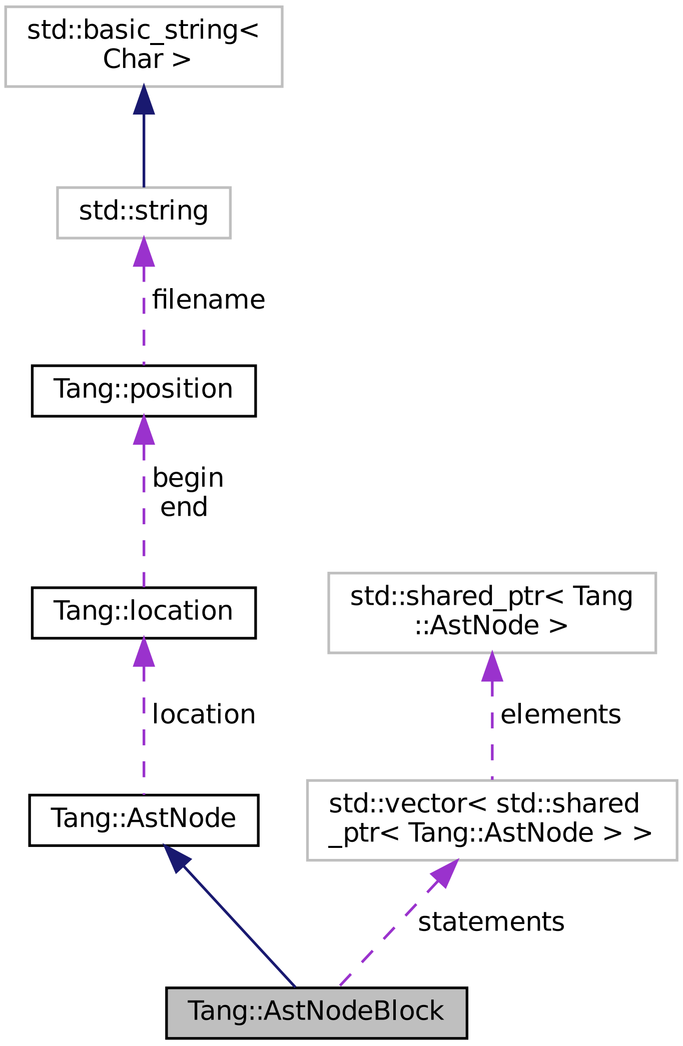
An AstNode that represents a code block.

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

Inheritance diagram for Tang::AstNodeBlock:



Collaboration diagram for Tang::AstNodeBlock:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- std::vector< std::shared_ptr< AstNode > > statements

  The statements included in the code block.
- Tang::location location

  The location associated with this node.

5.5.1 Detailed Description

An AstNode that represents a code block.

5.5.2 Member Enumeration Documentation

5.5.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.5.3 Constructor & Destructor Documentation

5.5.3.1 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.

5.5.4 Member Function Documentation

5.5.4.1 compile()

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

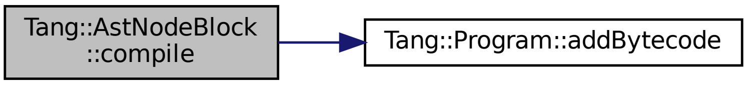
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.5.4.2 compilePreprocess()

```
void AstNodeBlock::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.5.4.3 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:

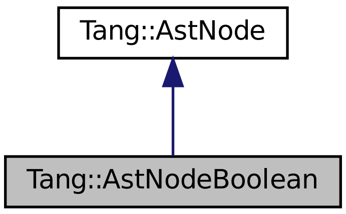
- include/astNodeBlock.hpp
- src/astNodeBlock.cpp

5.6 Tang::AstNodeBoolean Class Reference

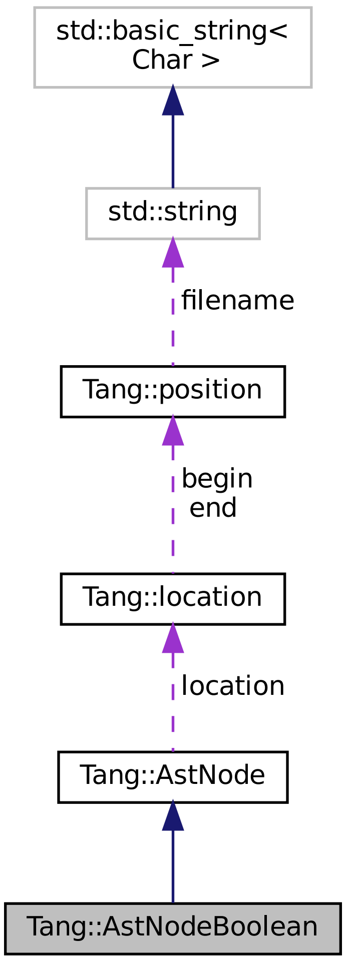
An AstNode that represents a boolean literal.

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

Inheritance diagram for Tang::AstNodeBoolean:



Collaboration diagram for Tang::AstNodeBoolean:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const

  Run any preprocess analysis needed before compilation.

Private Attributes

- bool val

    The boolean value being stored.
- Tang::location location

    The location associated with this node.

5.6.1 Detailed Description

An AstNode that represents a boolean literal.

5.6.2 Member Enumeration Documentation

5.6.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.6.3 Constructor & Destructor Documentation

5.6.3.1 AstNodeBoolean()

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

The constructor.

Parameters

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

5.6.4 Member Function Documentation

5.6.4.1 compile()

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

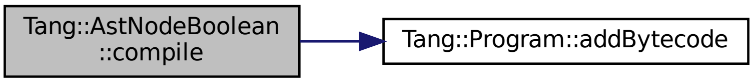
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.6.4.2 compilePreprocess()

```
void AstNode::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [virtual], [inherited]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeSlice, Tang::AstNodeReturn, Tang::AstNodeRangedFor, Tang::AstNodePrint, Tang::AstNodeIndex, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBlock, Tang::AstNodeBinary, Tang::AstNodeAssign, and Tang::AstNodeArray.

5.6.4.3 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:

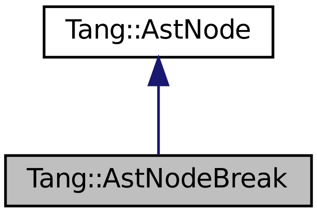
- include/astNodeBoolean.hpp
- src/astNodeBoolean.cpp

5.7 Tang::AstNodeBreak Class Reference

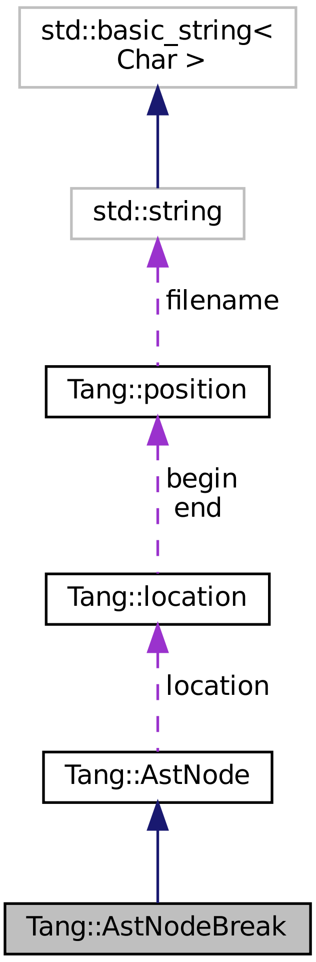
An AstNode that represents a `break` statement.

`#include <astNodeBreak.hpp>`

Inheritance diagram for Tang::AstNodeBreak:



Collaboration diagram for Tang::AstNodeBreak:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeBreak (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 compilePreprocess (Program &program, PreprocessState state) const

  Run any preprocess analysis needed before compilation.

Private Attributes

- Tang::location location

  The location associated with this node.

5.7.1 Detailed Description

An AstNode that represents a `break` statement.

5.7.2 Member Enumeration Documentation

5.7.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.7.3 Constructor & Destructor Documentation

5.7.3.1 AstNodeBreak()

```
AstNodeBreak::AstNodeBreak (
            Tang::location location )
```

The constructor.

Parameters

location	The location associated with the expression.

5.7.4 Member Function Documentation

5.7.4.1 compile()

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

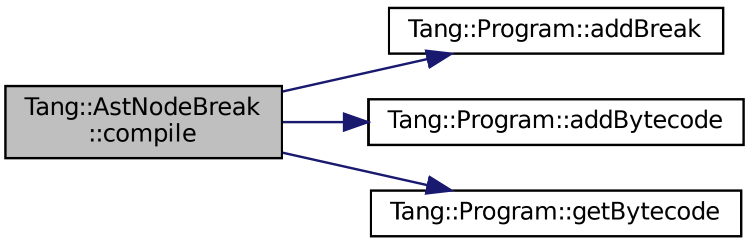
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeBreak ::compile
Tang::Program::addBreak
Tang::Program::addBytecode
Tang::Program::getBytecode

5.7.4.2 compilePreprocess()

```
void AstNode::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [virtual], [inherited]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeSlice, Tang::AstNodeReturn, Tang::AstNodeRangedFor, Tang::AstNodePrint, Tang::AstNodeIndex, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBlock, Tang::AstNodeBinary, Tang::AstNodeAssign, and Tang::AstNodeArray.

5.7.4.3 dump()

```
string AstNodeBreak::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:

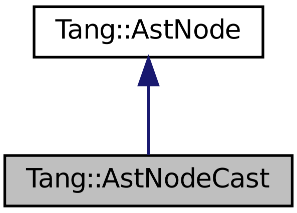
- include/astNodeBreak.hpp
- src/astNodeBreak.cpp

5.8 Tang::AstNodeCast Class Reference

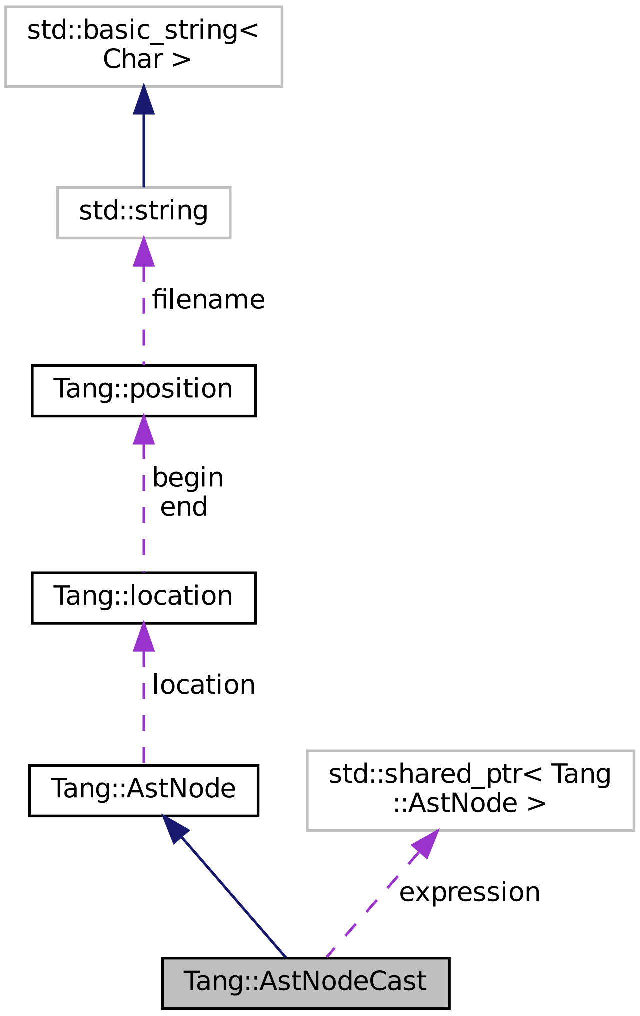
An AstNode that represents a typecast of an expression.

`#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.
- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- Type targetType

  The target type.
- shared_ptr< AstNode > expression

  The expression being typecast.
- Tang::location location

  The location associated with this node.

5.8.1 Detailed Description

An AstNode that represents a typecast of an expression.

5.8.2 Member Enumeration Documentation

5.8.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.8.2.2 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.

5.8.3 Constructor & Destructor Documentation

5.8.3.1 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.

5.8.4 Member Function Documentation

5.8.4.1 compile()

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

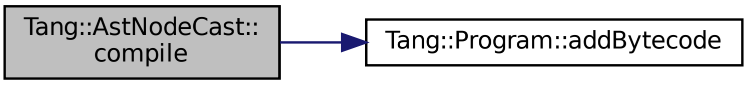
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.8.4.2 compilePreprocess()

```
void AstNodeCast::compilePreprocess (
             Program & program,
             PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.8.4.3 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:

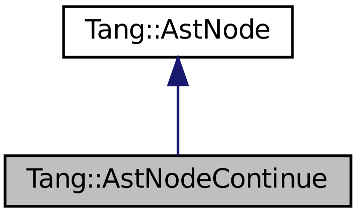
- include/astNodeCast.hpp
- src/astNodeCast.cpp

5.9 Tang::AstNodeContinue Class Reference

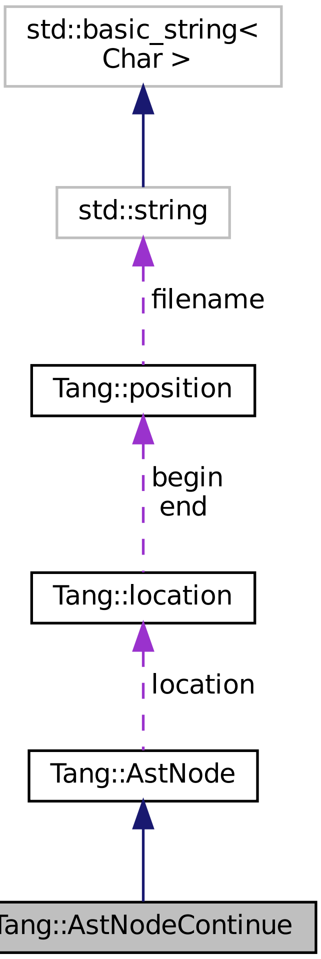
An AstNode that represents a `continue` statement.

```
#include <astNodeContinue.hpp>
```

Inheritance diagram for Tang::AstNodeContinue:



Collaboration diagram for Tang::AstNodeContinue:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

    Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeContinue (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 compilePreprocess (Program &program, PreprocessState state) const

    Run any preprocess analysis needed before compilation.

Private Attributes

- Tang::location location

    The location associated with this node.

5.9.1 Detailed Description

An AstNode that represents a `continue` statement.

5.9.2 Member Enumeration Documentation

5.9.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.9.3 Constructor & Destructor Documentation

5.9.3.1 AstNodeContinue()

```
AstNodeContinue::AstNodeContinue (
            Tang::location location )
```

The constructor.

Parameters

location	The location associated with the expression.

5.9.4 Member Function Documentation

5.9.4.1 compile()

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

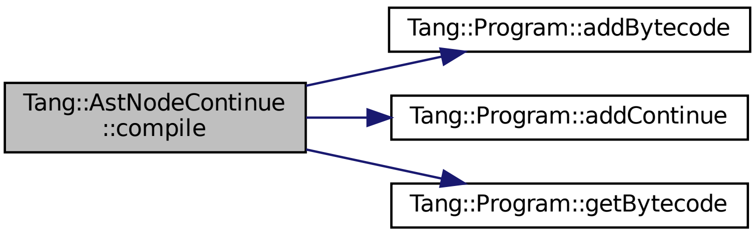
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeContinue ::compile
Tang::Program::addBytecode
Tang::Program::addContinue
Tang::Program::getBytecode

5.9.4.2 compilePreprocess()

```
void AstNode::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [virtual], [inherited]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeSlice, Tang::AstNodeReturn, Tang::AstNodeRangedFor, Tang::AstNodePrint, Tang::AstNodeIndex, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBlock, Tang::AstNodeBinary, Tang::AstNodeAssign, and Tang::AstNodeArray.

5.9.4.3 dump()

```
string AstNodeContinue::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:

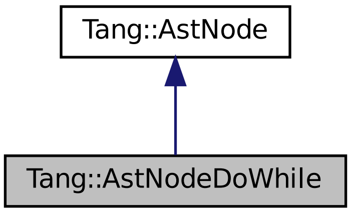
- include/astNodeContinue.hpp
- src/astNodeContinue.cpp

5.10 Tang::AstNodeDoWhile Class Reference

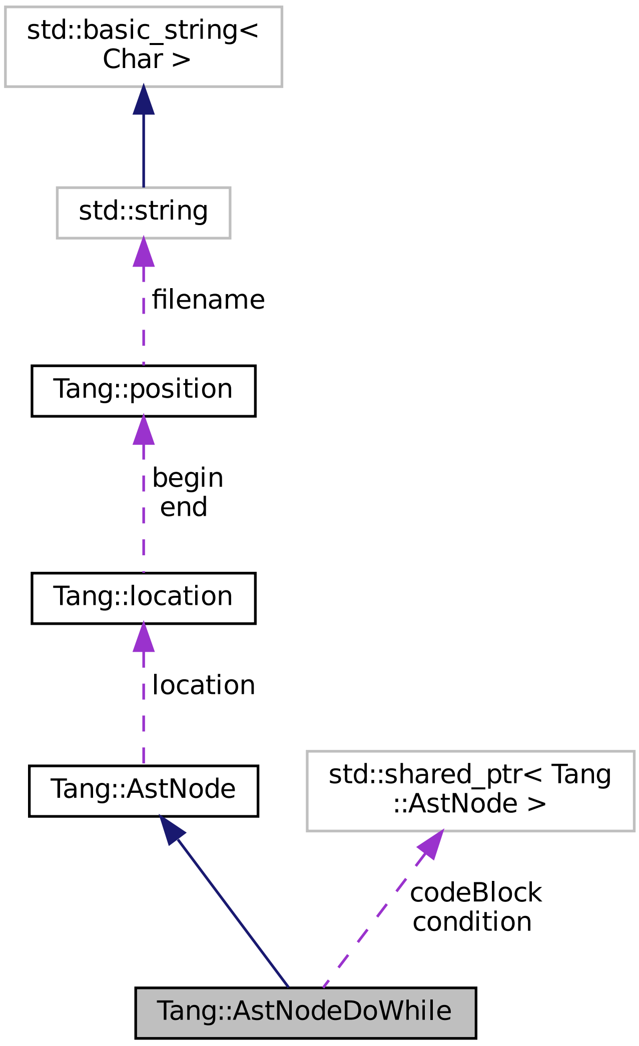
An AstNode that represents a do..while statement.

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

Inheritance diagram for Tang::AstNodeDoWhile:



Collaboration diagram for Tang::AstNodeDoWhile:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- shared_ptr< AstNode > condition

  The expression which determines whether or not the code block will continue to be executed.
- shared_ptr< AstNode > codeBlock

  The code block executed when the condition is true.
- Tang::location location

  The location associated with this node.

5.10.1 Detailed Description

An AstNode that represents a do..while statement.

5.10.2 Member Enumeration Documentation

5.10.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.10.3 Constructor & Destructor Documentation

5.10.3.1 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.

5.10.4 Member Function Documentation

5.10.4.1 compile()

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

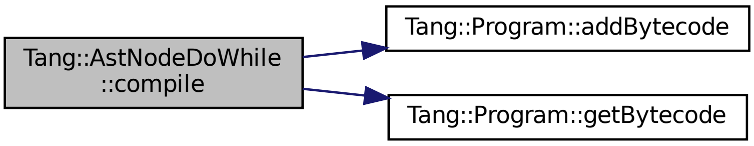
Compile the ast of the provided Tang::Program.

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

5.10.4.2 compilePreprocess()

```
void AstNodeDoWhile::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.10.4.3 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:

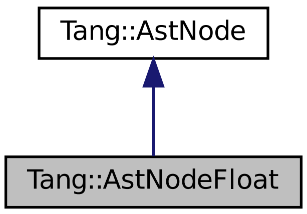
- include/astNodeDoWhile.hpp
- src/astNodeDoWhile.cpp

5.11 Tang::AstNodeFloat Class Reference

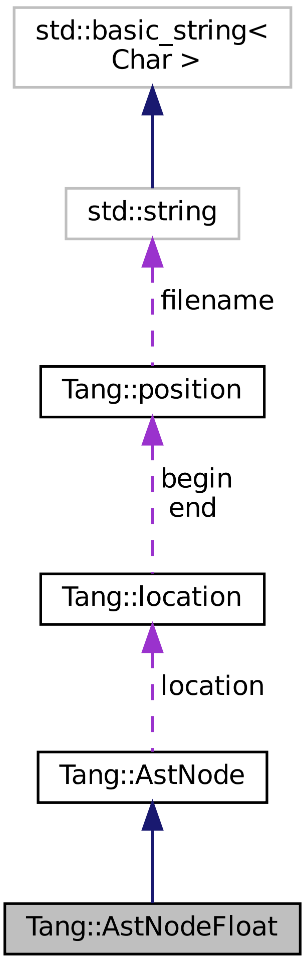
An AstNode that represents an float literal.

`#include <astNodeFloat.hpp>`

Inheritance diagram for Tang::AstNodeFloat:



Collaboration diagram for Tang::AstNodeFloat:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeFloat (Tang::float_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 compilePreprocess (Program &program, PreprocessState state) const

  Run any preprocess analysis needed before compilation.

Private Attributes

- Tang::float_t val

    The float value being stored.
- Tang::location location

    The location associated with this node.

5.11.1 Detailed Description

An AstNode that represents an float literal.

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

5.11.2 Member Enumeration Documentation

5.11.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.11.3 Constructor & Destructor Documentation

5.11.3.1 AstNodeFloat()

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

The constructor.

Parameters

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

5.11.4 Member Function Documentation

5.11.4.1 compile()

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

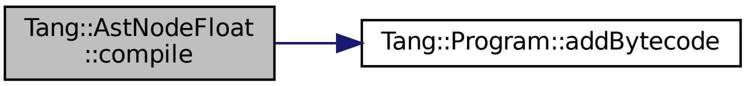
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.11.4.2 compilePreprocess()

```
void AstNode::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [virtual], [inherited]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeSlice, Tang::AstNodeReturn, Tang::AstNodeRangedFor, Tang::AstNodePrint, Tang::AstNodeIndex, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBlock, Tang::AstNodeBinary, Tang::AstNodeAssign, and Tang::AstNodeArray.

5.11.4.3 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:

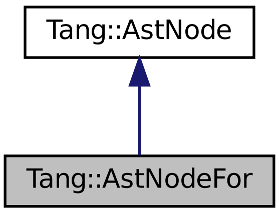
- include/astNodeFloat.hpp
- src/astNodeFloat.cpp

5.12 Tang::AstNodeFor Class Reference

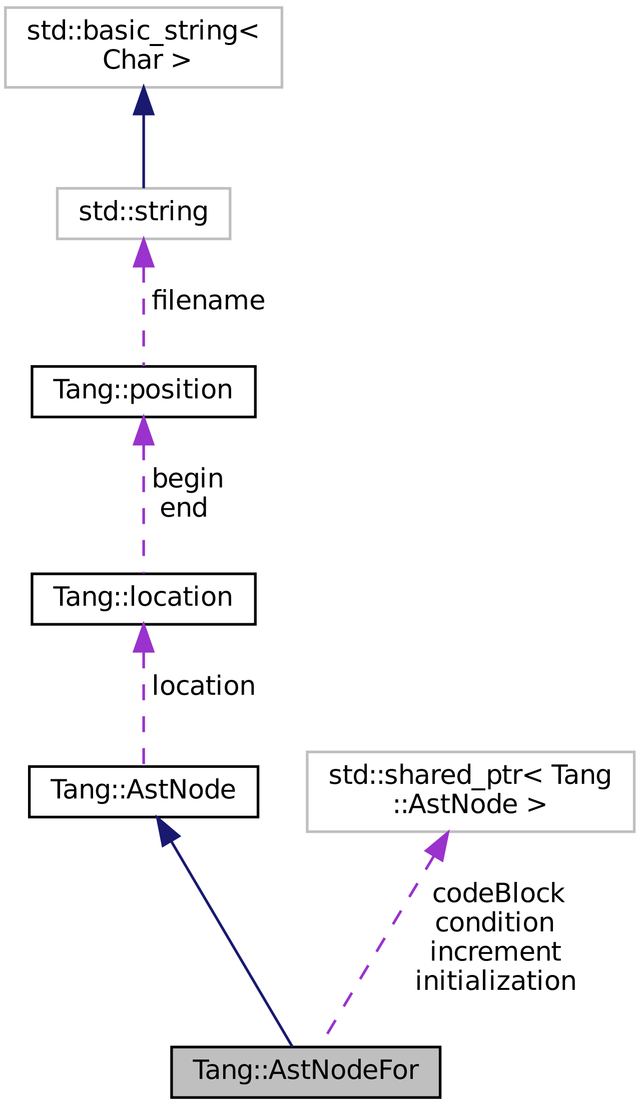
An AstNode that represents an if() statement.

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

Inheritance diagram for Tang::AstNodeFor:



Collaboration diagram for Tang::AstNodeFor:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- shared_ptr< AstNode > initialization

  The expression to be executed first to set up the for() loop.
- shared_ptr< AstNode > condition

  The expression which determines whether or not the code block will continue to be executed.
- shared_ptr< AstNode > increment

  The expression to be executed immediately after the code block.
- shared_ptr< AstNode > codeBlock

  The code block executed when the condition is true.
- Tang::location location

  The location associated with this node.

5.12.1 Detailed Description

An AstNode that represents an if() statement.

5.12.2 Member Enumeration Documentation

5.12.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.12.3 Constructor & Destructor Documentation

5.12.3.1 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.

5.12.4 Member Function Documentation

5.12.4.1 compile()

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

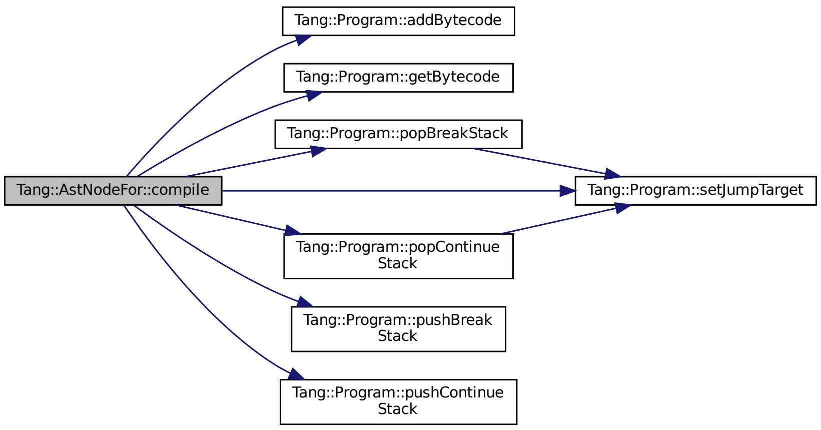
Compile the ast of the provided Tang::Program.

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::popBreakStack
Tang::Program::setJumpTarget
Tang::Program::popContinue Stack
Tang::Program::pushBreak Stack
Tang::Program::pushContinue Stack

5.12.4.2 compilePreprocess()

```
void AstNodeFor::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.12.4.3 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:

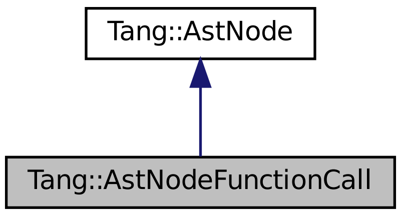
- include/astNodeFor.hpp
- src/astNodeFor.cpp

5.13 Tang::AstNodeFunctionCall Class Reference

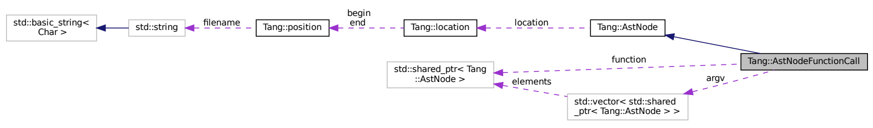
An AstNode that represents a function call.

`#include <astNodeFunctionCall.hpp>`

Inheritance diagram for Tang::AstNodeFunctionCall:



Collaboration diagram for Tang::AstNodeFunctionCall:


std::basic_string< Char >
std::string
filename
Tang::position
begin
end
Tang::location
location
Tang::AstNode
Tang::AstNodeFunctionCall
function
std::shared_ptr< Tang ::AstNode >
elements
argv
std::vector< std::shared _ptr< Tang::AstNode > >

Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeFunctionCall (std::shared_ptr< AstNode > function, std::vector< std::shared_ptr< AstNode >> argv, 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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- std::shared_ptr< AstNode > function

  The function being invoked.
- std::vector< std::shared_ptr< AstNode > > argv

  The list of arguments provided to the function.
- Tang::location location

  The location associated with this node.

5.13.1 Detailed Description

An AstNode that represents a function call.

5.13.2 Member Enumeration Documentation

5.13.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.13.3 Constructor & Destructor Documentation

5.13.3.1 AstNodeFunctionCall()

```
AstNodeFunctionCall::AstNodeFunctionCall (
            std::shared_ptr< AstNode > function,
            std::vector< std::shared_ptr< AstNode >> argv,
            Tang::location location )
```

The constructor.

Parameters

function	The function being invoked.
argv	The list of arguments provided to the function.
location	The location associated with the expression.

5.13.4 Member Function Documentation

5.13.4.1 compile()

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

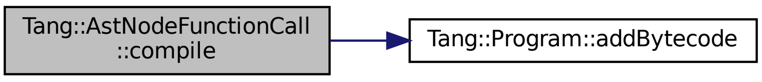
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeFunctionCall ::compile
Tang::Program::addBytecode

5.13.4.2 compilePreprocess()

```
void AstNodeFunctionCall::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.13.4.3 dump()

```
string AstNodeFunctionCall::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:

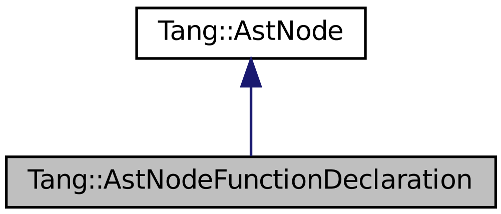
- include/astNodeFunctionCall.hpp
- src/astNodeFunctionCall.cpp

5.14 Tang::AstNodeFunctionDeclaration Class Reference

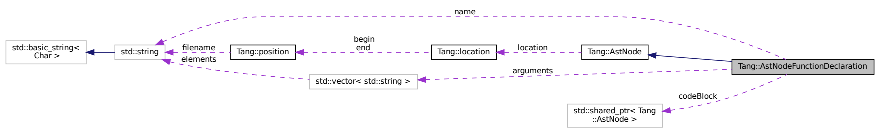
An AstNode that represents a function declaration.

`#include <astNodeFunctionDeclaration.hpp>`

Inheritance diagram for Tang::AstNodeFunctionDeclaration:



Collaboration diagram for Tang::AstNodeFunctionDeclaration:


name
std::basic_string< Char >
std::string
filename
elements
Tang::position
begin
end
Tang::location
location
Tang::AstNode
Tang::AstNodeFunctionDeclaration
arguments
std::vector< std::string >
codeBlock
std::shared_ptr< Tang ::AstNode >

Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

    Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeFunctionDeclaration (std::string name, std::vector< std::string > arguments, 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 compilePreprocess (Program &program, PreprocessState state) const override

    Run any preprocess analysis needed before compilation.

Private Attributes

- std::string name

    The name of the function.
- std::vector< std::string > arguments

    The arguments expected to be provided.
- shared_ptr< AstNode > codeBlock

    The code block executed when the condition is true.
- Tang::location location

    The location associated with this node.

5.14.1 Detailed Description

An AstNode that represents a function declaration.

5.14.2 Member Enumeration Documentation

5.14.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.14.3 Constructor & Destructor Documentation

5.14.3.1 AstNodeFunctionDeclaration()

```
AstNodeFunctionDeclaration::AstNodeFunctionDeclaration (
            std::string name,
            std::vector< std::string > arguments,
            shared_ptr< AstNode > codeBlock,
            Tang::location location )
```

The constructor.

Parameters

name	The name of the function.
arguments	The arguments expected to be provided.
codeBlock	The code executed as part of the function.
location	The location associated with the function declaration.

5.14.4 Member Function Documentation

5.14.4.1 compile()

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

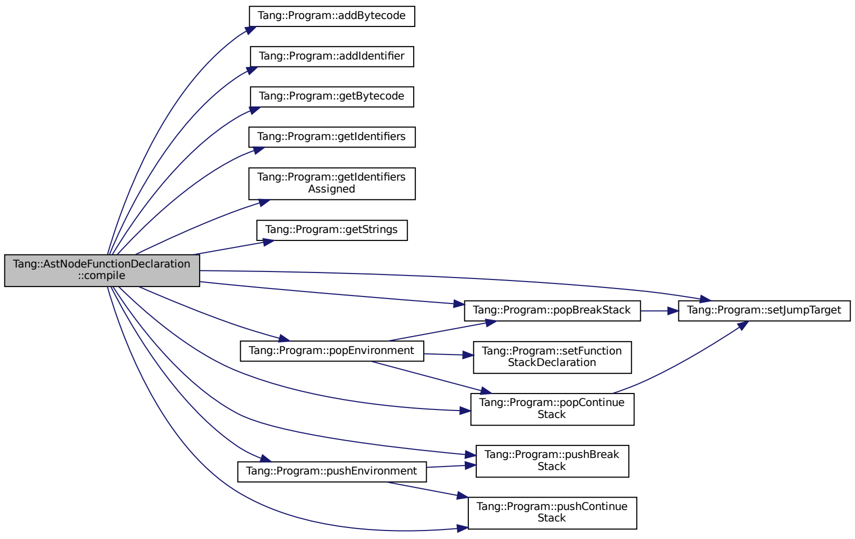
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::Program::addBytecode
Tang::Program::addIdentifier
Tang::Program::getBytecode
Tang::Program::getIdentifiers
Tang::Program::getIdentifiers Assigned
Tang::Program::getStrings
Tang::AstNodeFunctionDeclaration ::compile
Tang::Program::popBreakStack
Tang::Program::setJumpTarget
Tang::Program::popEnvironment
Tang::Program::setFunction StackDeclaration
Tang::Program::popContinue Stack
Tang::Program::pushBreak Stack
Tang::Program::pushEnvironment
Tang::Program::pushContinue Stack

5.14.4.2 compilePreprocess()

```
void AstNodeFunctionDeclaration::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

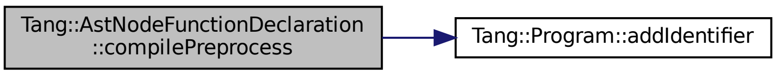
Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeFunctionDeclaration ::compilePreprocess
Tang::Program::addIdentifier

5.14.4.3 dump()

```
string AstNodeFunctionDeclaration::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:

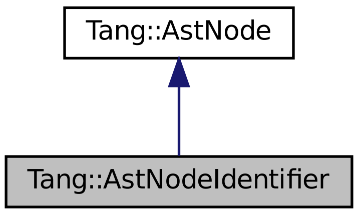
- include/astNodeFunctionDeclaration.hpp
- src/astNodeFunctionDeclaration.cpp

5.15 Tang::AstNodeIdentifier Class Reference

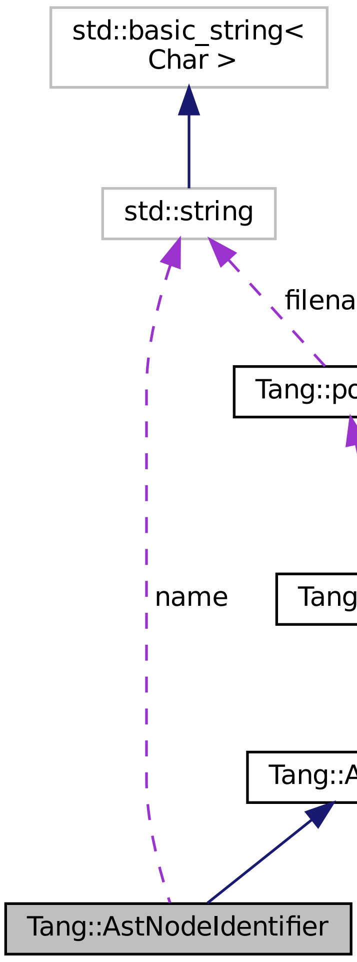
An AstNode that represents an identifier.

`#include <astNodeIdentifier.hpp>`

Inheritance diagram for Tang::AstNodeIdentifier:



Collaboration diagram for Tang::AstNodeIdentifier:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Public Attributes

- std::string name

  The name of the identifier.

Private Attributes

- Tang::location location

  The location associated with this node.

5.15.1 Detailed Description

An AstNode that represents an identifier.

Identifier names are represented by a string.

5.15.2 Member Enumeration Documentation

5.15.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.15.3 Constructor & Destructor Documentation

5.15.3.1 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.

5.15.4 Member Function Documentation

5.15.4.1 compile()

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

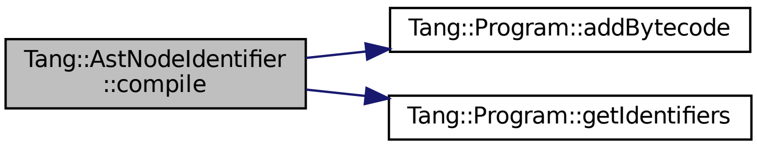
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeIdentifier ::compile
Tang::Program::addBytecode
Tang::Program::getIdentifiers

5.15.4.2 compilePreprocess()

```
void AstNodeIdentifier::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

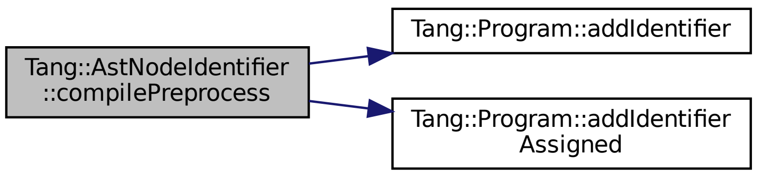
Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeIdentifier ::compilePreprocess
Tang::Program::addIdentifier
Tang::Program::addIdentifier Assigned

5.15.4.3 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:

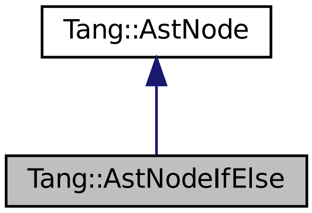
- include/astNodeIdentifier.hpp
- src/astNodeIdentifier.cpp

5.16 Tang::AstNodeIfElse Class Reference

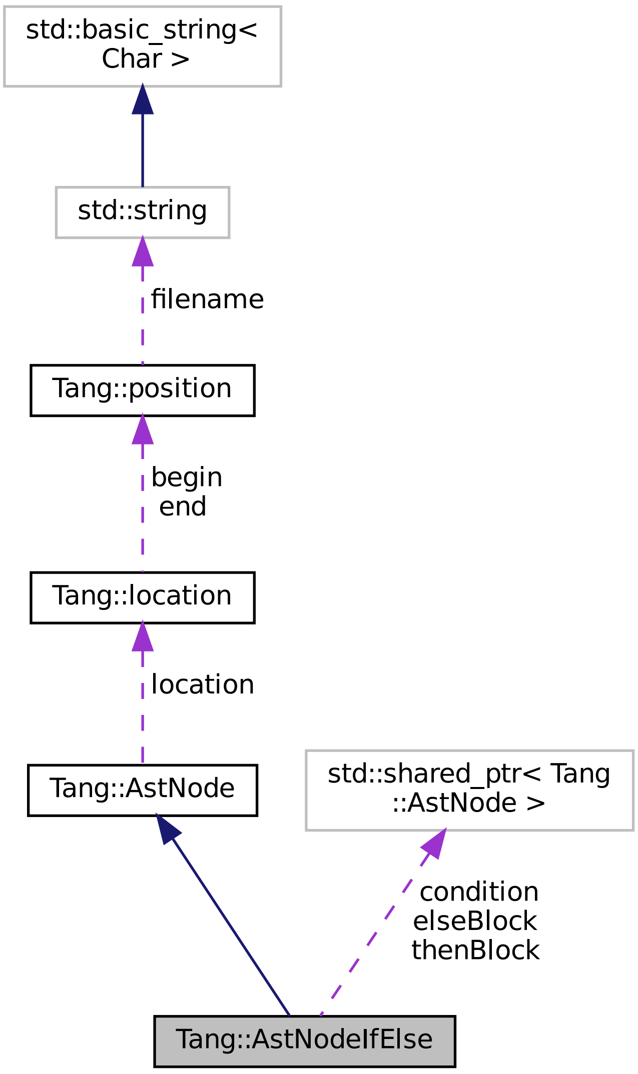
An AstNode that represents an if..else statement.

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

Inheritance diagram for Tang::AstNodeIfElse:



Collaboration diagram for Tang::AstNodeIfElse:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- shared_ptr< AstNode > condition

  The expression which determines whether the thenBlock or elseBlock is executed.
- shared_ptr< AstNode > thenBlock

  The statement executed when the condition is true.
- shared_ptr< AstNode > elseBlock

  The statement executed when the condition is false.
- Tang::location location

  The location associated with this node.

5.16.1 Detailed Description

An AstNode that represents an if..else statement.

5.16.2 Member Enumeration Documentation

5.16.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.16.3 Constructor & Destructor Documentation

5.16.3.1 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.

5.16.3.2 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.

5.16.4 Member Function Documentation

5.16.4.1 compile()

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

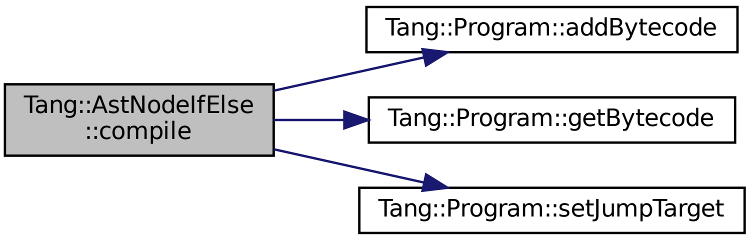
Compile the ast of the provided Tang::Program.

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

5.16.4.2 compilePreprocess()

```
void AstNodeIfElse::compilePreprocess (
             Program & program,
             PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.16.4.3 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:

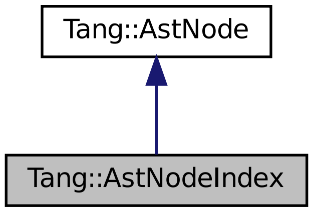
- include/astNodeIfElse.hpp
- src/astNodeIfElse.cpp

5.17 Tang::AstNodeIndex Class Reference

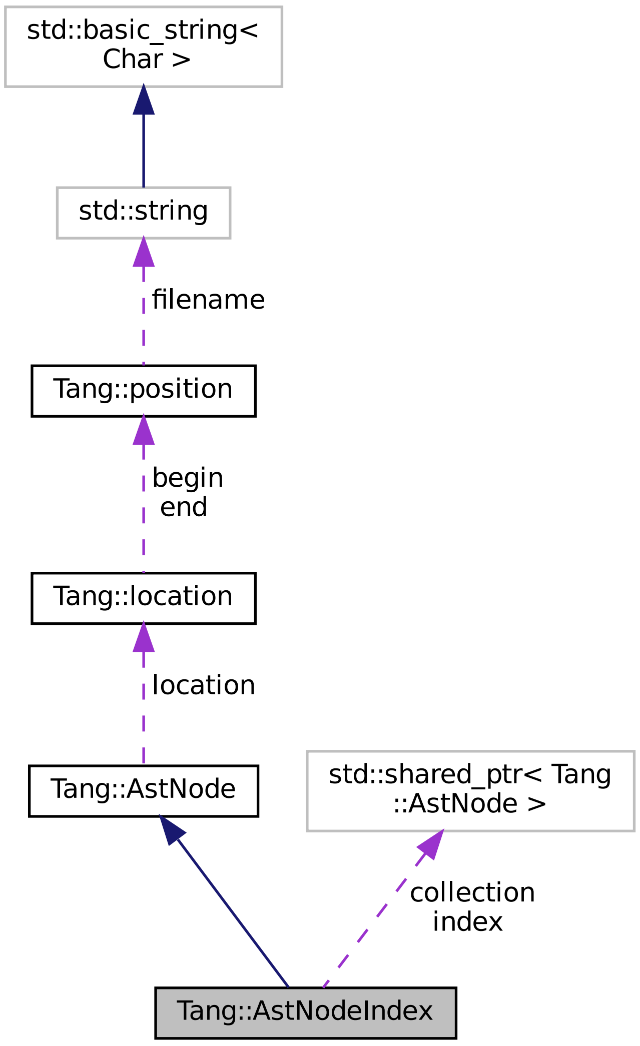
An AstNode that represents an index into a collection.

`#include <astNodeIndex.hpp>`

Inheritance diagram for Tang::AstNodeIndex:



Collaboration diagram for Tang::AstNodeIndex:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

    Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeIndex (std::shared_ptr< AstNode > collection, std::shared_ptr< AstNode > index, 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 compilePreprocess (Program &program, PreprocessState state) const override

    Run any preprocess analysis needed before compilation.
- const std::shared_ptr< const AstNode > getCollection () const

    Return a shared pointer to the AstNode serving as the Collection.
- const std::shared_ptr< const AstNode > getIndex () const

    Return a shared pointer to the AstNode serving as the Index.

Private Attributes

- std::shared_ptr< AstNode > collection

    The collection into which we will index.
- std::shared_ptr< AstNode > index

    The index expression.
- Tang::location location

    The location associated with this node.

5.17.1 Detailed Description

An AstNode that represents an index into a collection.

5.17.2 Member Enumeration Documentation

5.17.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.17.3 Constructor & Destructor Documentation

5.17.3.1 AstNodeIndex()

```
AstNodeIndex::AstNodeIndex (
            std::shared_ptr< AstNode > collection,
            std::shared_ptr< AstNode > index,
            Tang::location location )
```

The constructor.

Parameters

collection	The collection into which we will index.
index	The index expression.
location	The location associated with the expression.

5.17.4 Member Function Documentation

5.17.4.1 compile()

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

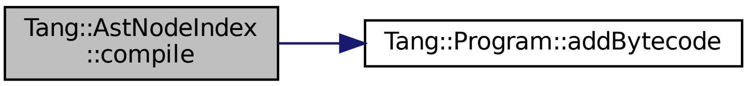
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.17.4.2 compilePreprocess()

```
void AstNodeIndex::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.17.4.3 dump()

```
string AstNodeIndex::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.

5.17.4.4 getCollection()

```
const std::shared_ptr< const AstNode > AstNodeIndex::getCollection ( ) const
```

Return a shared pointer to the AstNode serving as the Collection.

Returns

The collection into which we will index.

5.17.4.5 getIndex()

```
const std::shared_ptr< const AstNode > AstNodeIndex::getIndex ( ) const
```

Return a shared pointer to the AstNode serving as the Index.

Returns

The index expression.

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

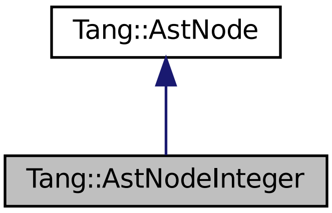
- include/astNodeIndex.hpp
- src/astNodeIndex.cpp

5.18 Tang::AstNodeInteger Class Reference

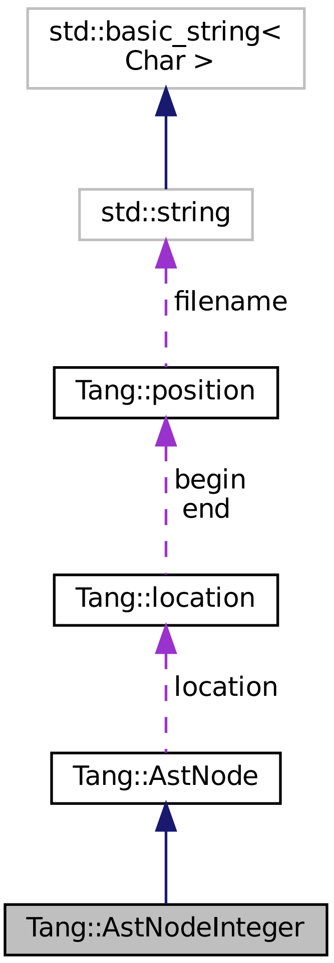
An AstNode that represents an integer literal.

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

Inheritance diagram for Tang::AstNodeInteger:



Collaboration diagram for Tang::AstNodeInteger:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

    Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeInteger (Tang::integer_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 compilePreprocess (Program &program, PreprocessState state) const

    Run any preprocess analysis needed before compilation.

Private Attributes

- Tang::integer_t val

  The integer value being stored.
- Tang::location location

  The location associated with this node.

5.18.1 Detailed Description

An AstNode that represents an integer literal.

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

5.18.2 Member Enumeration Documentation

5.18.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.18.3 Constructor & Destructor Documentation

5.18.3.1 AstNodeInteger()

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

The constructor.

Parameters

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

5.18.4 Member Function Documentation

5.18.4.1 compile()

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

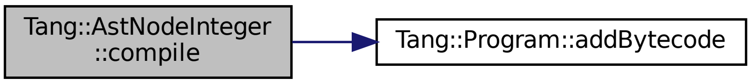
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.18.4.2 compilePreprocess()

```
void AstNode::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [virtual], [inherited]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeTernary, Tang::AstNodeString, Tang::AstNodeSlice, Tang::AstNodeReturn, Tang::AstNodeRangedFor, Tang::AstNodePrint, Tang::AstNodeIndex, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFunctionDeclaration, Tang::AstNodeFunctionCall, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBlock, Tang::AstNodeBinary, Tang::AstNodeAssign, and Tang::AstNodeArray.

5.18.4.3 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:

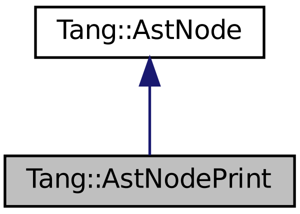
- include/astNodeInteger.hpp
- src/astNodeInteger.cpp

5.19 Tang::AstNodePrint Class Reference

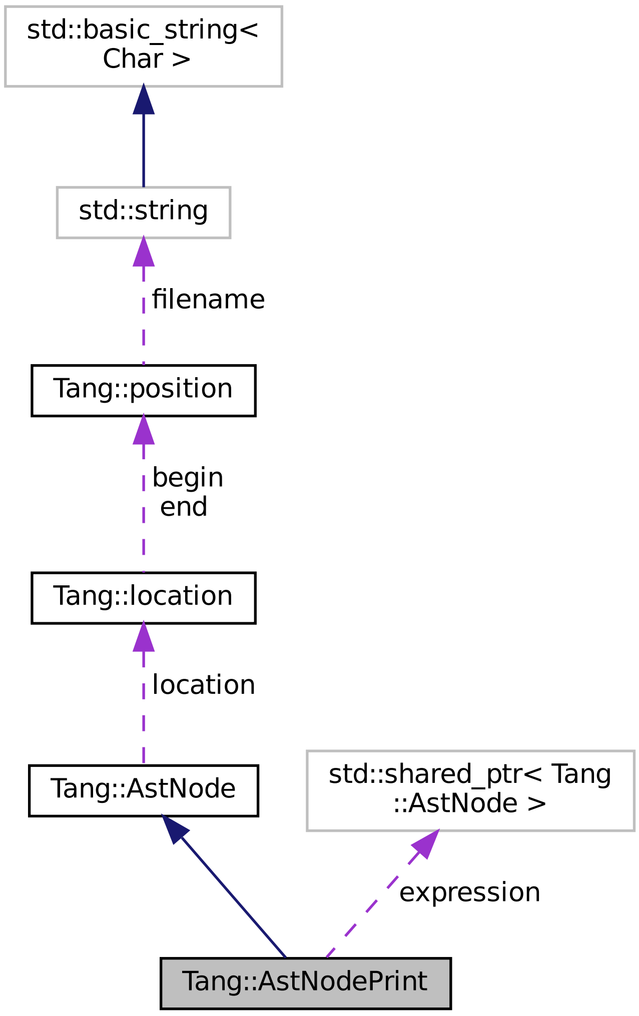
An AstNode that represents a print typeeration.

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

Inheritance diagram for Tang::AstNodePrint:



Collaboration diagram for Tang::AstNodePrint:



Public Types

- enum Type { Default }

    The type of print() requested.
- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

    Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- Type type

  The type of print() being requested.
- shared_ptr< AstNode > expression

  The expression to be printed.
- Tang::location location

  The location associated with this node.

5.19.1 Detailed Description

An AstNode that represents a print typeeration.

5.19.2 Member Enumeration Documentation

5.19.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.19.2.2 Type

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

The type of print() requested.

Enumerator

Default	Use the default print.

5.19.3 Constructor & Destructor Documentation

5.19.3.1 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.

5.19.4 Member Function Documentation

5.19.4.1 compile()

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

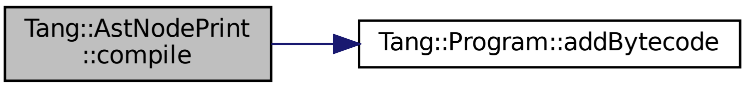
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.19.4.2 compilePreprocess()

```
void AstNodePrint::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.19.4.3 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:

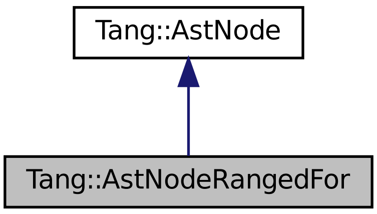
- include/astNodePrint.hpp
- src/astNodePrint.cpp

5.20 Tang::AstNodeRangedFor Class Reference

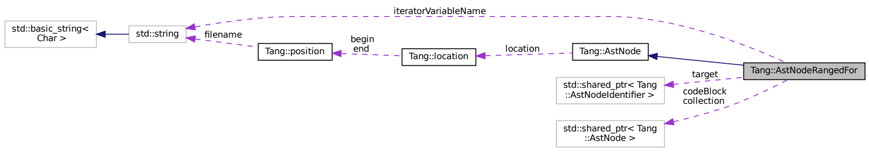
An AstNode that represents a ranged for() statement.

```
#include <astNodeRangedFor.hpp>
```

Inheritance diagram for Tang::AstNodeRangedFor:



Collaboration diagram for Tang::AstNodeRangedFor:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

    Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeRangedFor (shared_ptr< AstNodeIdentifier > target, shared_ptr< AstNode > collection, 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 compilePreprocess (Program &program, PreprocessState state) const override

    Run any preprocess analysis needed before compilation.

Private Attributes

- shared_ptr< AstNodeIdentifier > target

    The target variable to hold the value for the current loop iteration.
- shared_ptr< AstNode > collection

    The collection through which to iterate.
- shared_ptr< AstNode > codeBlock

    The code block executed when the condition is true.
- string iteratorVariableName

    The unique variable name that this iterator will use to persist its state on the stack.
- Tang::location location

    The location associated with this node.

5.20.1 Detailed Description

An AstNode that represents a ranged for() statement.

5.20.2 Member Enumeration Documentation

5.20.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.20.3 Constructor & Destructor Documentation

5.20.3.1 AstNodeRangedFor()

```
AstNodeRangedFor::AstNodeRangedFor (
            shared_ptr< AstNodeIdentifier > target,
            shared_ptr< AstNode > collection,
            shared_ptr< AstNode > codeBlock,
            Tang::location location )
```

The constructor.

Parameters

target	The target variable to hold the value for the current loop iteration.
collection	The collection through which to iterate.
codeBlock	The statement executed when the condition is true.
location	The location associated with the expression.

5.20.4 Member Function Documentation

5.20.4.1 compile()

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

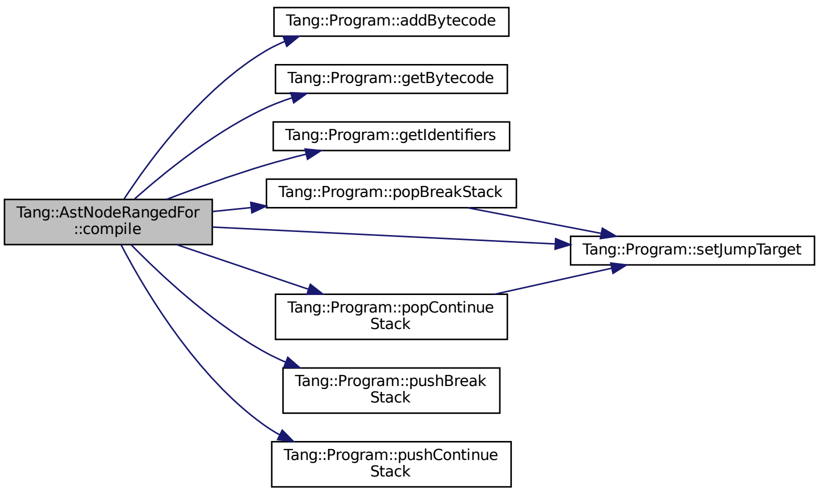
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::Program::addBytecode
Tang::Program::getBytecode
Tang::Program::getIdentifiers
Tang::Program::popBreakStack
Tang::AstNodeRangedFor ::compile
Tang::Program::setJumpTarget
Tang::Program::popContinue Stack
Tang::Program::pushBreak Stack
Tang::Program::pushContinue Stack

5.20.4.2 compilePreprocess()

```
void AstNodeRangedFor::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

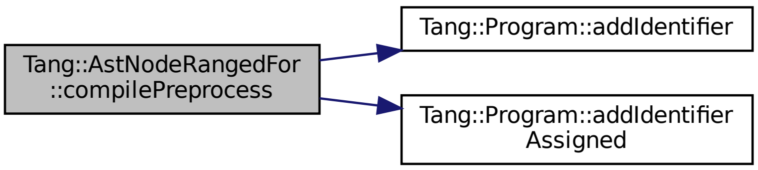
Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeRangedFor ::compilePreprocess
Tang::Program::addIdentifier
Tang::Program::addIdentifier Assigned

5.20.4.3 dump()

```
string AstNodeRangedFor::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:

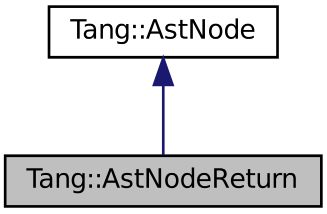
- include/astNodeRangedFor.hpp
- src/astNodeRangedFor.cpp

5.21 Tang::AstNodeReturn Class Reference

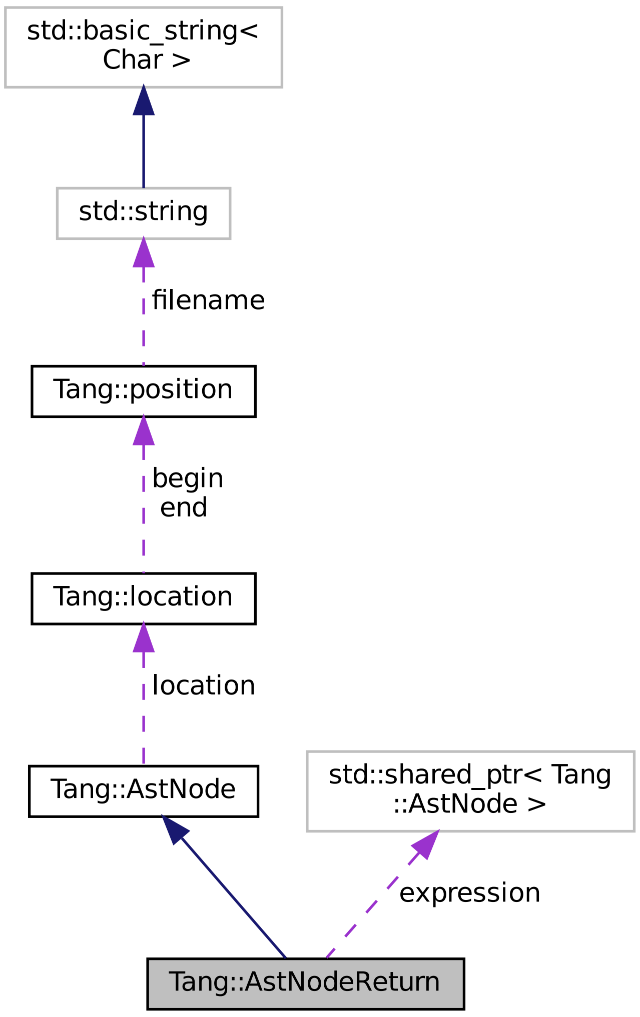
An AstNode that represents a `return` statement.

`#include <astNodeReturn.hpp>`

Inheritance diagram for Tang::AstNodeReturn:



Collaboration diagram for Tang::AstNodeReturn:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeReturn (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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- shared_ptr< AstNode > expression

  The expression to which the operation will be applied.
- Tang::location location

  The location associated with this node.

5.21.1 Detailed Description

An AstNode that represents a `return` statement.

5.21.2 Member Enumeration Documentation

5.21.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.21.3 Constructor & Destructor Documentation

5.21.3.1 AstNodeReturn()

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

The constructor.

Parameters

expression	The expression to be returned.
location	The location associated with the return statement.

5.21.4 Member Function Documentation

5.21.4.1 compile()

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

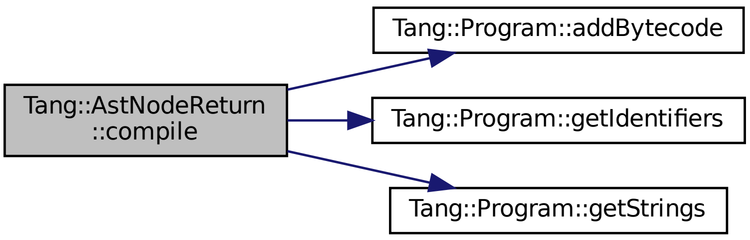
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeReturn ::compile
Tang::Program::addBytecode
Tang::Program::getIdentifiers
Tang::Program::getStrings

5.21.4.2 compilePreprocess()

```
void AstNodeReturn::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.21.4.3 dump()

```
string AstNodeReturn::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:

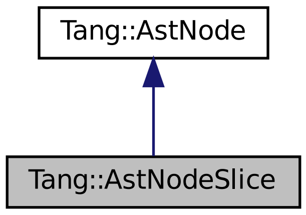
- include/astNodeReturn.hpp
- src/astNodeReturn.cpp

5.22 Tang::AstNodeSlice Class Reference

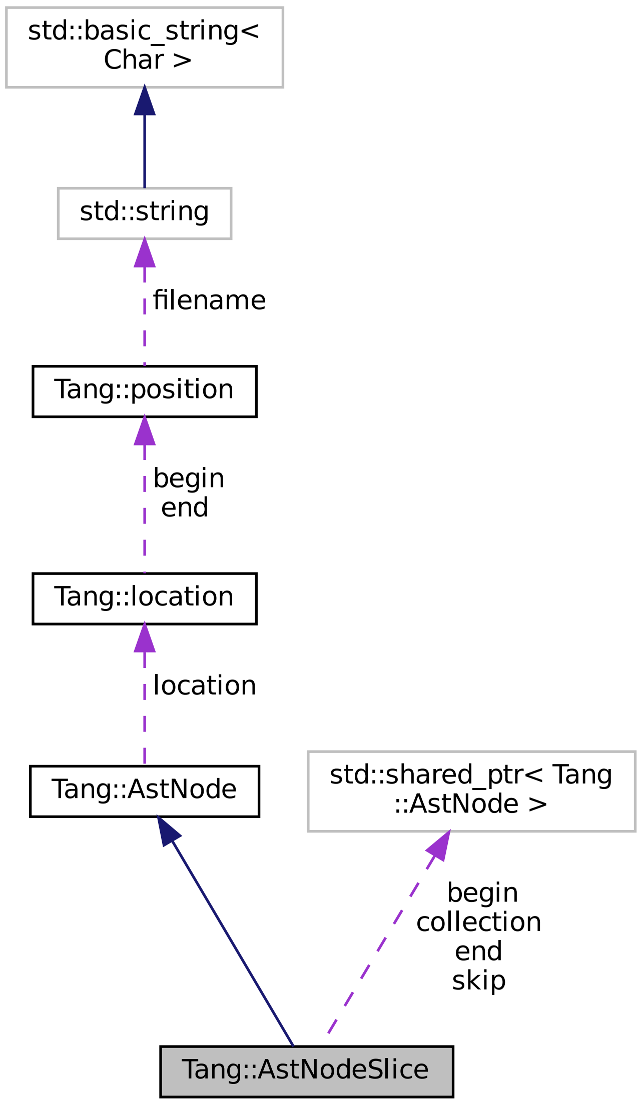
An AstNode that represents a ternary expression.

```
#include <astNodeSlice.hpp>
```

Inheritance diagram for Tang::AstNodeSlice:



Collaboration diagram for Tang::AstNodeSlice:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

    Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Public Member Functions

- AstNodeSlice (shared_ptr< AstNode > collection, shared_ptr< AstNode > begin, shared_ptr< AstNode > end, shared_ptr< AstNode > slice, 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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- shared_ptr< AstNode > collection

  The collection which will be sliced.
- shared_ptr< AstNode > begin

  The begin index position of the slice.
- shared_ptr< AstNode > end

  The end index position of the slice.
- shared_ptr< AstNode > skip

  The skip index position of the slice.
- Tang::location location

  The location associated with this node.

5.22.1 Detailed Description

An AstNode that represents a ternary expression.

5.22.2 Member Enumeration Documentation

5.22.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.22.3 Constructor & Destructor Documentation

5.22.3.1 AstNodeSlice()

```
AstNodeSlice::AstNodeSlice (
             shared_ptr< AstNode > collection,
             shared_ptr< AstNode > begin,
             shared_ptr< AstNode > end,
             shared_ptr< AstNode > slice,
             Tang::location location )
```

The constructor.

Parameters

collection	The collection which will be sliced.
begin	The begin index position of the slice.
end	The end index position of the slice.
skip	The skip index position of the slice.
location	The location associated with the expression.

5.22.4 Member Function Documentation

5.22.4.1 compile()

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

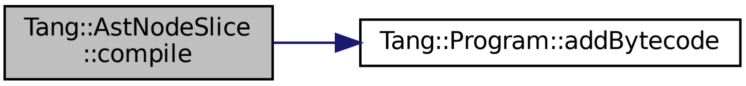
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.22.4.2 compilePreprocess()

```
void AstNodeSlice::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.22.4.3 dump()

```
string AstNodeSlice::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:

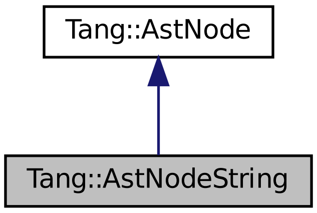
- include/astNodeSlice.hpp
- src/astNodeSlice.cpp

5.23 Tang::AstNodeString Class Reference

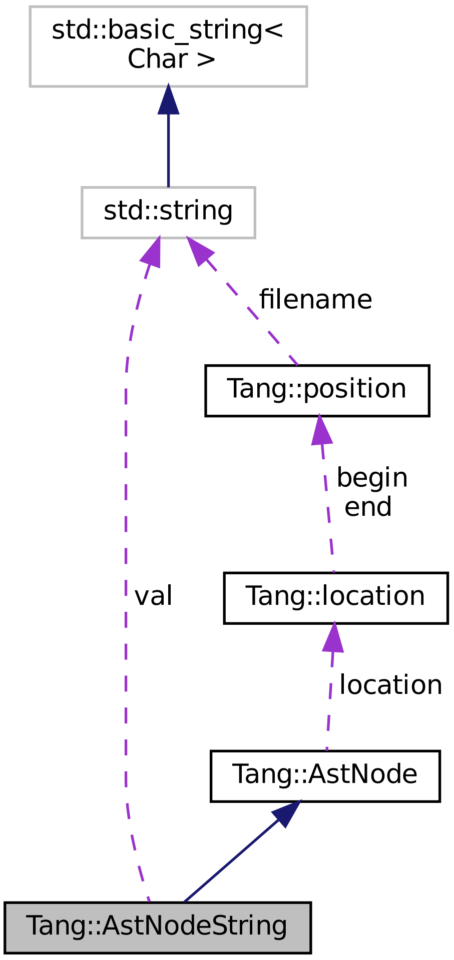
An AstNode that represents a string literal.

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

Inheritance diagram for Tang::AstNodeString:



Collaboration diagram for Tang::AstNodeString:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.
- void compileLiteral (Tang::Program &program) const

  Compile the string and push it onto the stack.

Private Attributes

- std::string val

  The string value being stored.
- Tang::location location

  The location associated with this node.

5.23.1 Detailed Description

An AstNode that represents a string literal.

5.23.2 Member Enumeration Documentation

5.23.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.23.3 Constructor & Destructor Documentation

5.23.3.1 AstNodeString()

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

The constructor.

Parameters

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

5.23.4 Member Function Documentation

5.23.4.1 compile()

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

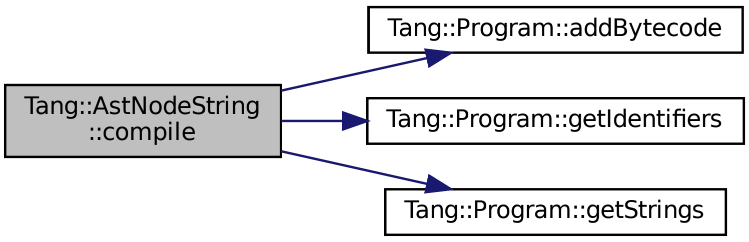
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::AstNodeString ::compile
Tang::Program::addBytecode
Tang::Program::getIdentifiers
Tang::Program::getStrings

5.23.4.2 compileLiteral()

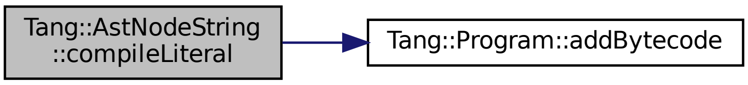
```
void AstNodeString::compileLiteral (
            Tang::Program & program ) const
```

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

5.23.4.3 compilePreprocess()

```
void AstNodeString::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

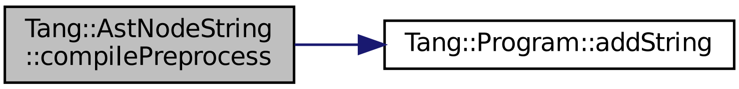
Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.23.4.4 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:

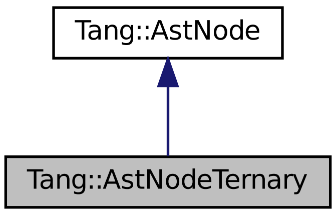
- include/astNodeString.hpp
- src/astNodeString.cpp

5.24 Tang::AstNodeTernary Class Reference

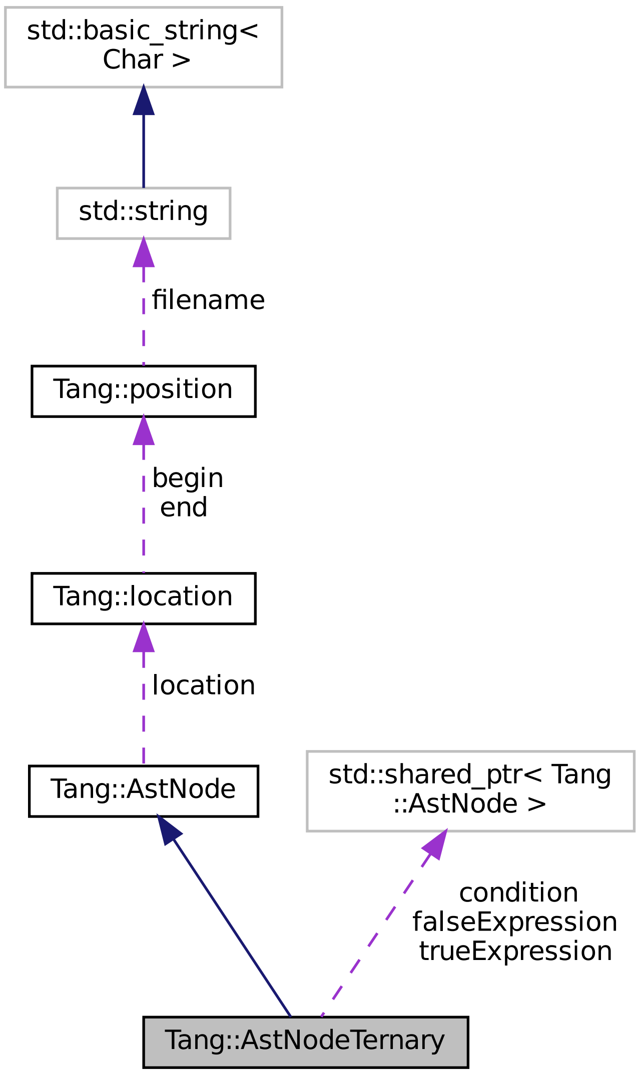
An AstNode that represents a ternary expression.

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

Inheritance diagram for Tang::AstNodeTernary:



Collaboration diagram for Tang::AstNodeTernary:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- shared_ptr< AstNode > condition

  The expression which determines whether the trueExpression or falseExpression is executed.

- shared_ptr< AstNode > trueExpression

  The expression executed when the condition is true.

- shared_ptr< AstNode > falseExpression

  The expression executed when the condition is false.

- Tang::location location

  The location associated with this node.

5.24.1 Detailed Description

An AstNode that represents a ternary expression.

5.24.2 Member Enumeration Documentation

5.24.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.24.3 Constructor & Destructor Documentation

5.24.3.1 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.

5.24.4 Member Function Documentation

5.24.4.1 compile()

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

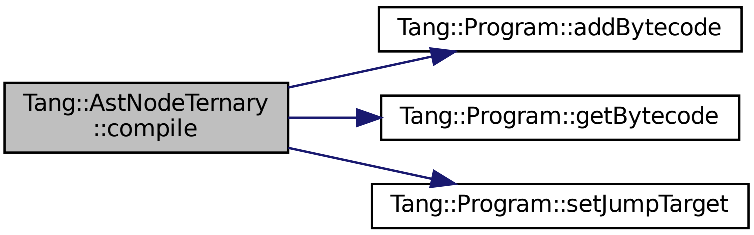
Compile the ast of the provided Tang::Program.

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

5.24.4.2 compilePreprocess()

```
void AstNodeTernary::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.24.4.3 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:

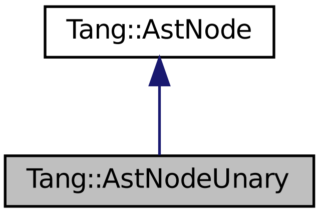
- include/astNodeTernary.hpp
- src/astNodeTernary.cpp

5.25 Tang::AstNodeUnary Class Reference

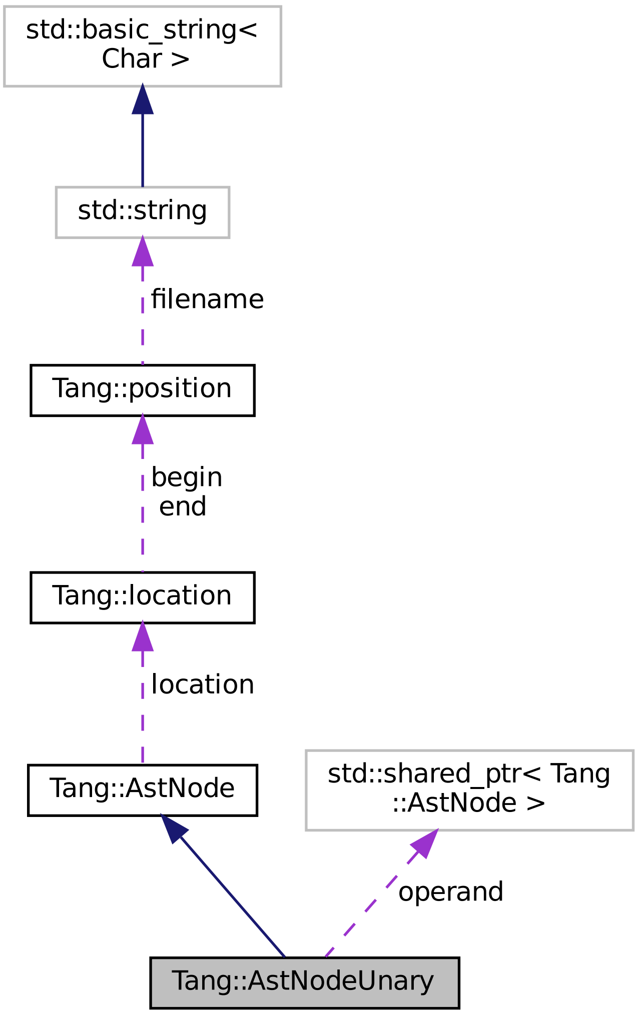
An AstNode that represents a unary negation.

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

Inheritance diagram for Tang::AstNodeUnary:



Collaboration diagram for Tang::AstNodeUnary:



Public Types

- enum Operator { Negative , Not }

  The type of operation.
- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const override

  Run any preprocess analysis needed before compilation.

Private Attributes

- Operator op

  The operation which will be applied to the operand.
- shared_ptr< AstNode > operand

  The operand to which the operation will be applied.
- Tang::location location

  The location associated with this node.

5.25.1 Detailed Description

An AstNode that represents a unary negation.

5.25.2 Member Enumeration Documentation

5.25.2.1 Operator

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

The type of operation.

Enumerator

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

5.25.2.2 PreprocessState

```
enum Tang::AstNode::PreprocessState : int [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.25.3 Constructor & Destructor Documentation

5.25.3.1 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.

5.25.4 Member Function Documentation

5.25.4.1 compile()

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

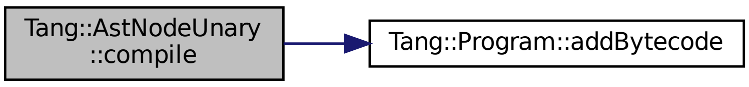
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.25.4.2 compilePreprocess()

```
void AstNodeUnary::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.25.4.3 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:

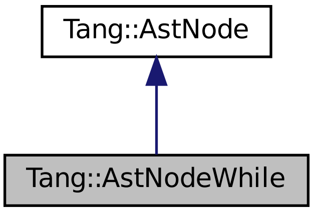
- include/astNodeUnary.hpp
- src/astNodeUnary.cpp

5.26 Tang::AstNodeWhile Class Reference

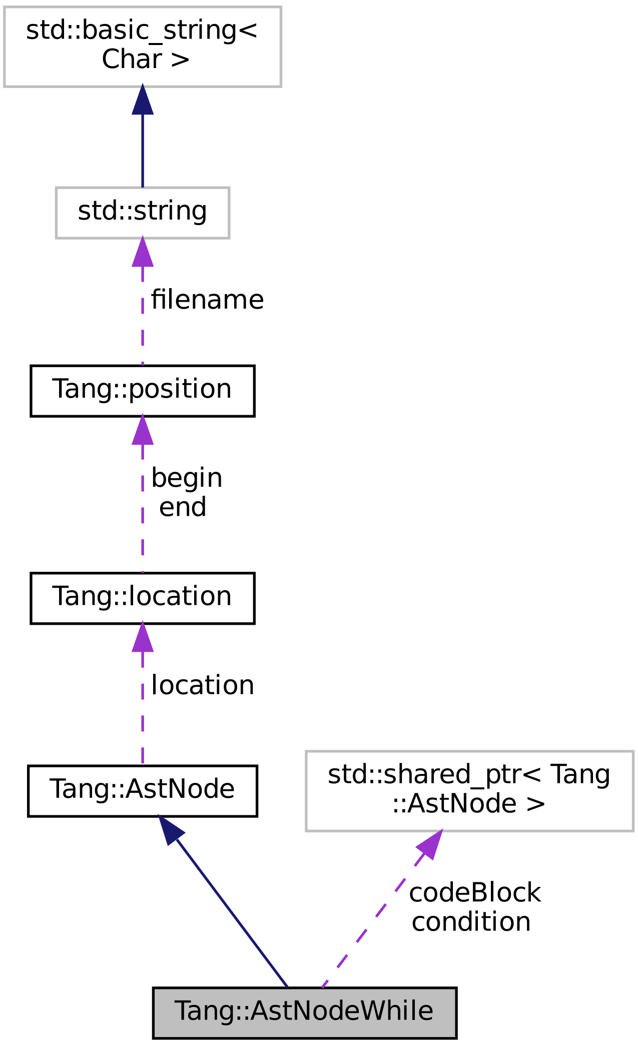
An AstNode that represents a while statement.

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

Inheritance diagram for Tang::AstNodeWhile:



Collaboration diagram for Tang::AstNodeWhile:



Public Types

- enum PreprocessState : int { Default = 0 , IsAssignment = 1 }

  Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

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 compilePreprocess (Program &program, PreprocessState state) const override

    Run any preprocess analysis needed before compilation.

Private Attributes

- shared_ptr< AstNode > condition

    The expression which determines whether or not the code block will continue to be executed.
- shared_ptr< AstNode > codeBlock

    The code block executed when the condition is true.
- Tang::location location

    The location associated with this node.

5.26.1 Detailed Description

An AstNode that represents a while statement.

5.26.2 Member Enumeration Documentation

5.26.2.1 PreprocessState

```
enum Tang::AstNode::PreprocessState :  int  [inherited]
```

Bit flags to indicate the state of the preprocess scan as it recursively evaluates the AST.

Enumerator

Default	The default state.
IsAssignment	AstNode is part of an assignment expression.

5.26.3 Constructor & Destructor Documentation

5.26.3.1 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.

5.26.4 Member Function Documentation

5.26.4.1 compile()

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

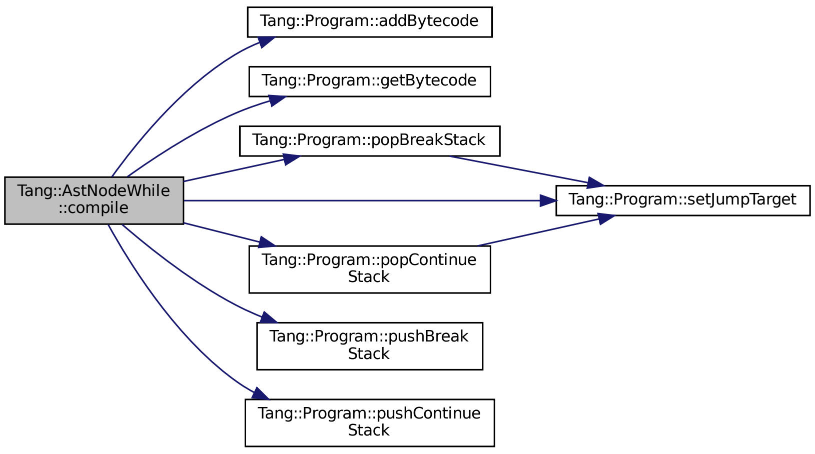
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:


Tang::Program::addBytecode
Tang::Program::getBytecode
Tang::Program::popBreakStack
Tang::AstNodeWhile ::compile
Tang::Program::setJumpTarget
Tang::Program::popContinue Stack
Tang::Program::pushBreak Stack
Tang::Program::pushContinue Stack

5.26.4.2 compilePreprocess()

```
void AstNodeWhile::compilePreprocess (
            Program & program,
            PreprocessState state ) const  [override], [virtual]
```

Run any preprocess analysis needed before compilation.

Parameters

program	The Tang::Program that is being compiled.
state	Any preprocess flags that need to be considered.

Reimplemented from Tang::AstNode.

5.26.4.3 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:

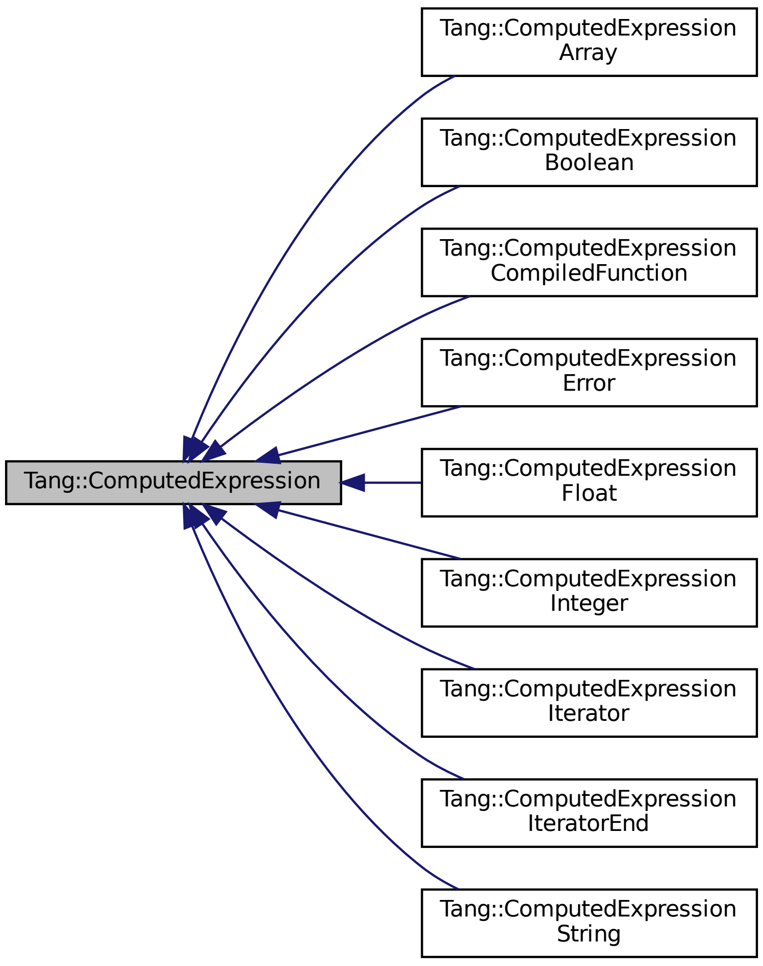
- include/astNodeWhile.hpp
- src/astNodeWhile.cpp

5.27 Tang::ComputedExpression Class Reference

Represents the result of a computation that has been executed.

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

Inheritance diagram for Tang::ComputedExpression:


Tang::ComputedExpressionArray
Tang::ComputedExpressionBoolean
Tang::ComputedExpressionCompiledFunction
Tang::ComputedExpressionError
Tang::ComputedExpression
Tang::ComputedExpressionFloat
Tang::ComputedExpressionInteger
Tang::ComputedExpressionIterator
Tang::ComputedExpressionIteratorEnd
Tang::ComputedExpressionString

Public Member Functions

- virtual ~ComputedExpression ()

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

  Output the contents of the ComputedExpression as a string.
- virtual std::string __asCode () const

  Output the contents of the ComputedExpression as a string similar to how it would be represented as code.
- virtual bool isCopyNeeded () const

  Determine whether or not a copy is needed.
- virtual GarbageCollected makeCopy () const

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

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)

  Perform an index assignment to the supplied 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 equality test.
- virtual GarbageCollected __index (const GarbageCollected &index) const

  Perform an index operation.
- virtual GarbageCollected __slice (const GarbageCollected &begin, const GarbageCollected &end, const GarbageCollected &skip) const

  Perform a slice operation.
- virtual GarbageCollected __getIterator (const GarbageCollected &collection) const

  Get an iterator for the expression.
- virtual GarbageCollected __iteratorNext (size_t index=0) const

  Get the next iterative value.
- virtual GarbageCollected __integer () const

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

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

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

  Perform a type cast to string.

5.27.1 Detailed Description

Represents the result of a computation that has been executed.

By default, it will represent a NULL value.

5.27.2 Member Function Documentation

5.27.2.1 __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.

5.27.2.2 __asCode()

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

Output the contents of the ComputedExpression as a string similar to how it would be represented as code.

Returns

A code-string representation of the computed expression.

Reimplemented in Tang::ComputedExpressionString.

5.27.2.3 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.27.2.4 __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.

5.27.2.5 __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.

5.27.2.6 __equal()

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

Perform an equality 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, Tang::ComputedExpressionCompiledFunction, and Tang::ComputedExpressionBoolean.

5.27.2.7 __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.

5.27.2.8 __getIterator()

```
GarbageCollected ComputedExpression::__getIterator (
             const GarbageCollected & collection ) const  [virtual]
```

Get an iterator for the expression.

Parameters

collection	The GarbageCollected value that will serve as the collection through which to iterate.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.27.2.9 __index()

```
GarbageCollected ComputedExpression::__index (
             const GarbageCollected & index ) const  [virtual]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.27.2.10 __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.

5.27.2.11 __iteratorNext()

```
GarbageCollected ComputedExpression::__iteratorNext (
             size_t index = 0 ) const  [virtual]
```

Get the next iterative value.

Parameters

index	The desired index value.

Reimplemented in Tang::ComputedExpressionString, Tang::ComputedExpressionIterator, and Tang::ComputedExpressionArray.

5.27.2.12 __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.

5.27.2.13 __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.

5.27.2.14 __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.

5.27.2.15 __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.

5.27.2.16 __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.

5.27.2.17 __slice()

```
GarbageCollected ComputedExpression::__slice (
             const GarbageCollected & begin,
             const GarbageCollected & end,
             const GarbageCollected & skip ) const  [virtual]
```

Perform a slice operation.

Convention will follow Python semantics, in which a slice will start at the provided index position, and go up to but not including the end index. The slice will default to an index increment of 1, but can be defined as another integer value.

Parameters

begin	The begin index expression provided by the script.
end	The end index expression provided by the script.
skip	The skip index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.27.2.18 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

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

5.27.2.19 __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.

5.27.2.20 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::ComputedExpressionIteratorEnd, Tang::ComputedExpressionIterator, Tang::ComputedExpressionInteger, Tang::ComputedExpressionFloat, Tang::ComputedExpressionError, Tang::ComputedExpressionC
Tang::ComputedExpressionBoolean, and Tang::ComputedExpressionArray.

5.27.2.21 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.

5.27.2.22 is_equal() [2/6]

```
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.

5.27.2.23 is_equal() [3/6]

```
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.

5.27.2.24 is_equal() [4/6]

```
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.

5.27.2.25 is_equal() [5/6]

```
virtual bool Tang::ComputedExpression::is_equal (
             const Tang::float_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.

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

5.27.2.26 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::integer_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.

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

5.27.2.27 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.27.2.28 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, Tang::ComputedExpressionCompiledFunction, Tang::ComputedExpressionBoolean, and Tang::ComputedExpressionArray.

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

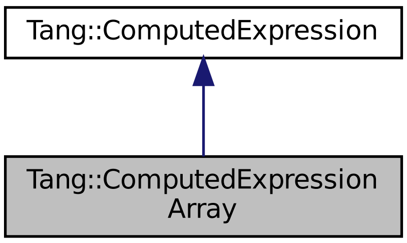
- include/computedExpression.hpp
- src/computedExpression.cpp

5.28 Tang::ComputedExpressionArray Class Reference

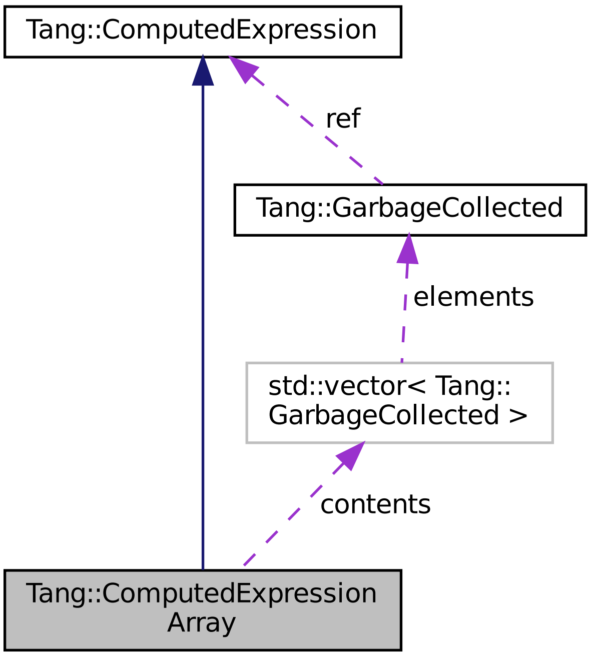
Represents an Array that is the result of a computation.

```
#include <computedExpressionArray.hpp>
```

Inheritance diagram for Tang::ComputedExpressionArray:



Collaboration diagram for Tang::ComputedExpressionArray:



Public Member Functions

- ComputedExpressionArray (std::vector< Tang::GarbageCollected > contents)

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

    Output the contents of the ComputedExpression as a string.
- virtual bool isCopyNeeded () const override

    Determine whether or not a copy is needed.
- GarbageCollected makeCopy () const override

    Make a copy of the ComputedExpression (recursively, if appropriate).
- virtual GarbageCollected __index (const GarbageCollected &index) const override

    Perform an index operation.
- virtual GarbageCollected __slice (const GarbageCollected &begin, const GarbageCollected &end, const GarbageCollected &skip) const override

    Perform a slice operation.
- virtual GarbageCollected __getIterator (const GarbageCollected &collection) const override

    Get an iterator for the expression.
- virtual GarbageCollected __iteratorNext (size_t index) const override

    Get the next iterative value.
- virtual GarbageCollected __assign_index (const GarbageCollected &index, const GarbageCollected &value) override

    Perform an index assignment to the supplied value.
- virtual GarbageCollected __string () const override

Perform a type cast to string.
- virtual std::string __asCode () const

  Output the contents of the ComputedExpression as a string similar to how it would be represented as code.
- virtual bool is_equal (const Tang::integer_t &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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 equality 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.

Private Attributes

- std::vector< Tang::GarbageCollected > contents

  The array contents.

5.28.1 Detailed Description

Represents an Array that is the result of a computation.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 ComputedExpressionArray()

```
ComputedExpressionArray::ComputedExpressionArray (
             std::vector< Tang::GarbageCollected > contents )
```

Construct an Array result.

Parameters

val	The integer value.

5.28.3 Member Function Documentation

5.28.3.1 __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.

5.28.3.2 __asCode()

```
string ComputedExpression::__asCode ( ) const  [virtual], [inherited]
```

Output the contents of the ComputedExpression as a string similar to how it would be represented as code.

Returns

A code-string representation of the computed expression.

Reimplemented in Tang::ComputedExpressionString.

5.28.3.3 __assign_index()

```
GarbageCollected ComputedExpressionArray::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value ) [override], [virtual]
```

Perform an index assignment to the supplied value.

Parameters

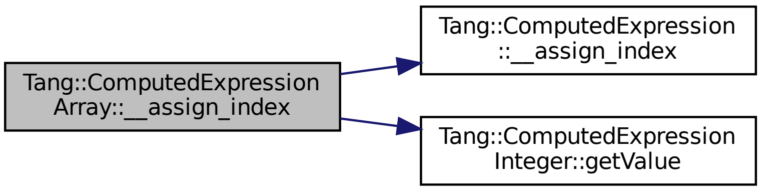
index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Array::__assign_index
Tang::ComputedExpression ::__assign_index
Tang::ComputedExpression Integer::getValue

5.28.3.4 __boolean()

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

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.

5.28.3.5 __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.

5.28.3.6 __equal()

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

Perform an equality 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, Tang::ComputedExpressionCompiledFunction, and Tang::ComputedExpressionBoolean.

5.28.3.7 __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.

5.28.3.8 __getIterator()

```
GarbageCollected ComputedExpressionArray::__getIterator (
            const GarbageCollected & collection ) const  [override], [virtual]
```

Get an iterator for the expression.

Parameters

collection	The GarbageCollected value that will serve as the collection through which to iterate.

Reimplemented from Tang::ComputedExpression.

5.28.3.9 __index()

```
GarbageCollected ComputedExpressionArray::__index (
            const GarbageCollected & index ) const  [override], [virtual]
```

Perform an index operation.

Parameters

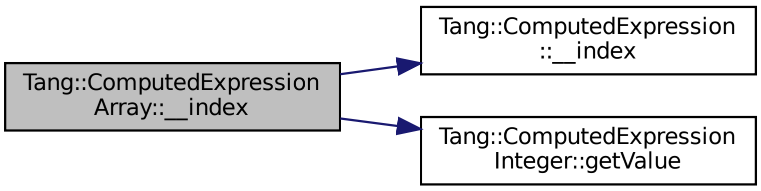
index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.28.3.10 __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.

5.28.3.11 __iteratorNext()

```
GarbageCollected ComputedExpressionArray::__iteratorNext (
            size_t index ) const  [override], [virtual]
```

Get the next iterative value.

Parameters

index	The desired index value.

Reimplemented from Tang::ComputedExpression.

5.28.3.12 __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.

5.28.3.13 __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.

5.28.3.14 __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.

5.28.3.15 __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.

5.28.3.16 __not()

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

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.

5.28.3.17 __slice()

```
GarbageCollected ComputedExpressionArray::__slice (
             const GarbageCollected & begin,
             const GarbageCollected & end,
             const GarbageCollected & skip ) const  [override], [virtual]
```

Perform a slice operation.

Convention will follow Python semantics, in which a slice will start at the provided index position, and go up to but not including the end index. The slice will default to an index increment of 1, but can be defined as another integer value.

Parameters

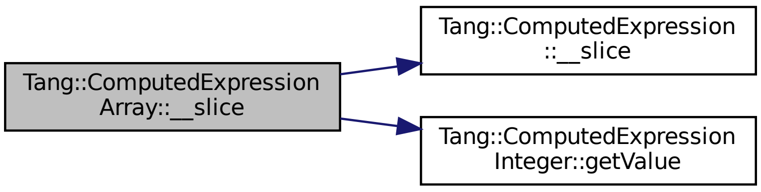
begin	The begin index expression provided by the script.
end	The end index expression provided by the script.
skip	The skip index expression provided by the script.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.28.3.18 __string()

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

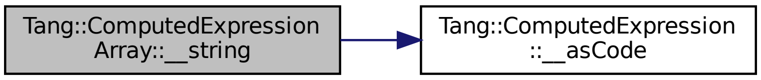
Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Array::__string
Tang::ComputedExpression ::__asCode

5.28.3.19 __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.

5.28.3.20 dump()

```
string ComputedExpressionArray::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.

5.28.3.21 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.

5.28.3.22 is_equal() [2/6]

```
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.

5.28.3.23 is_equal() [3/6]

```
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.

5.28.3.24 is_equal() [4/6]

```
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.

5.28.3.25 is_equal() [5/6]

```
virtual bool Tang::ComputedExpression::is_equal (
             const Tang::float_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.

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

5.28.3.26 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
             const Tang::integer_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.

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

5.28.3.27 isCopyNeeded()

```
bool ComputedExpressionArray::isCopyNeeded ( ) const  [override], [virtual]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented from Tang::ComputedExpression.

5.28.3.28 makeCopy()

```
GarbageCollected ComputedExpressionArray::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:

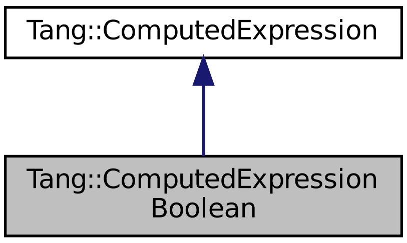
- include/computedExpressionArray.hpp
- src/computedExpressionArray.cpp

5.29 Tang::ComputedExpressionBoolean Class Reference

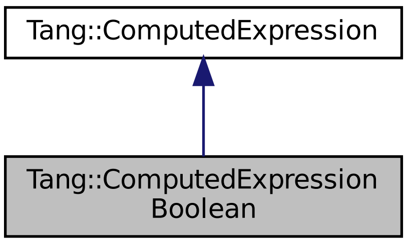
Represents an Boolean that is the result of a computation.

```
#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 equality 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 std::string __asCode () const

  Output the contents of the ComputedExpression as a string similar to how it would be represented as code.
- virtual bool isCopyNeeded () const

  Determine whether or not a copy is needed.
- virtual bool is_equal (const Tang::integer_t &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)

  Perform an index assignment to the supplied 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 __index (const GarbageCollected &index) const

  Perform an index operation.
- virtual GarbageCollected __slice (const GarbageCollected &begin, const GarbageCollected &end, const GarbageCollected &skip) const

  Perform a slice operation.
- virtual GarbageCollected __getIterator (const GarbageCollected &collection) const

  Get an iterator for the expression.
- virtual GarbageCollected __iteratorNext (size_t index=0) const

  Get the next iterative value.
- virtual GarbageCollected __string () const

  Perform a type cast to string.

Private Attributes

- bool val

    The boolean value.

5.29.1 Detailed Description

Represents an Boolean that is the result of a computation.

5.29.2 Constructor & Destructor Documentation

5.29.2.1 ComputedExpressionBoolean()

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

Construct an Boolean result.

Parameters

val	The boolean value.

5.29.3 Member Function Documentation

5.29.3.1 __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.

5.29.3.2 __asCode()

```
string ComputedExpression::__asCode ( ) const  [virtual], [inherited]
```

Output the contents of the ComputedExpression as a string similar to how it would be represented as code.

Returns

A code-string representation of the computed expression.

Reimplemented in Tang::ComputedExpressionString.

5.29.3.3 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
             const GarbageCollected & index,
             const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.29.3.4 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.29.3.5 __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.

5.29.3.6 __equal()

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

Perform an equality test.

Parameters

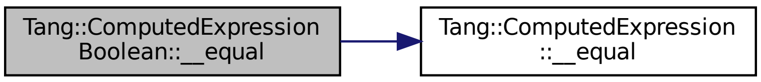
rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.29.3.7 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.29.3.8 __getIterator()

```
GarbageCollected ComputedExpression::__getIterator (
            const GarbageCollected & collection ) const  [virtual], [inherited]
```

Get an iterator for the expression.

Parameters

collection	The GarbageCollected value that will serve as the collection through which to iterate.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.29.3.9 __index()

```
GarbageCollected ComputedExpression::__index (
            const GarbageCollected & index ) const  [virtual], [inherited]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.29.3.10 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.29.3.11 __iteratorNext()

```
GarbageCollected ComputedExpression::__iteratorNext (
            size_t index = 0 ) const  [virtual], [inherited]
```

Get the next iterative value.

Parameters

index	The desired index value.

Reimplemented in Tang::ComputedExpressionString, Tang::ComputedExpressionIterator, and Tang::ComputedExpressionArray.

5.29.3.12 __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.

5.29.3.13 __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.

5.29.3.14 __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.

5.29.3.15 __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.

5.29.3.16 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.29.3.17 __slice()

```
GarbageCollected ComputedExpression::__slice (
            const GarbageCollected & begin,
            const GarbageCollected & end,
            const GarbageCollected & skip ) const  [virtual], [inherited]
```

Perform a slice operation.

Convention will follow Python semantics, in which a slice will start at the provided index position, and go up to but not including the end index. The slice will default to an index increment of 1, but can be defined as another integer value.

Parameters

begin	The begin index expression provided by the script.
end	The end index expression provided by the script.
skip	The skip index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.29.3.18 __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::ComputedExpressionIteratorEnd, Tang::ComputedExpressionInteger, Tang::ComputedExpressionFloat, Tang::ComputedExpressionError, and Tang::ComputedExpressionArray.

5.29.3.19 __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.

5.29.3.20 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.

5.29.3.21 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.

5.29.3.22 is_equal() [2/6]

```
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.

5.29.3.23 is_equal() [3/6]

```
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.

5.29.3.24 is_equal() [4/6]

```
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.

5.29.3.25 is_equal() [5/6]

```
virtual bool Tang::ComputedExpression::is_equal (
             const Tang::float_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.

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

5.29.3.26 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::integer_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.

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

5.29.3.27 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.29.3.28 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:

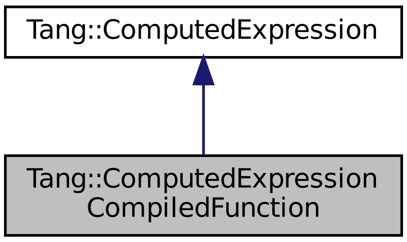
- include/computedExpressionBoolean.hpp
- src/computedExpressionBoolean.cpp

5.30 Tang::ComputedExpressionCompiledFunction Class Reference

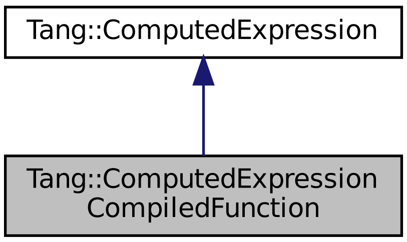
Represents a Compiled Function declared in the script.

```
#include <computedExpressionCompiledFunction.hpp>
```

Inheritance diagram for Tang::ComputedExpressionCompiledFunction:



Collaboration diagram for Tang::ComputedExpressionCompiledFunction:



Public Member Functions

- ComputedExpressionCompiledFunction (uint32_t argc, Tang::integer_t pc)

    Construct an CompiledFunction.
- 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 GarbageCollected __equal (const GarbageCollected &rhs) const override

    Perform an equality test.
- uint32_t getArgc () const

    Get the `argc` value.

- Tang::integer_t getPc () const

    Get the bytecode target.
- virtual std::string __asCode () const

    Output the contents of the ComputedExpression as a string similar to how it would be represented as code.
- virtual bool isCopyNeeded () const

    Determine whether or not a copy is needed.
- virtual bool is_equal (const Tang::integer_t &val) const

    Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)

    Perform an index assignment to the supplied 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 __index (const GarbageCollected &index) const

    Perform an index operation.
- virtual GarbageCollected __slice (const GarbageCollected &begin, const GarbageCollected &end, const GarbageCollected &skip) const

    Perform a slice operation.
- virtual GarbageCollected __getIterator (const GarbageCollected &collection) const

    Get an iterator for the expression.
- virtual GarbageCollected __iteratorNext (size_t index=0) const

    Get the next iterative value.
- virtual GarbageCollected __integer () const

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

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

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

    Perform a type cast to string.

Private Attributes

- uint32_t argc

    The count of arguments that this function expects.
- Tang::integer_t pc

    The bytecode addres of the start of the function.

5.30.1 Detailed Description

Represents a Compiled Function declared in the script.

5.30.2 Constructor & Destructor Documentation

5.30.2.1 ComputedExpressionCompiledFunction()

```
ComputedExpressionCompiledFunction::ComputedExpressionCompiledFunction (
             uint32_t argc,
             Tang::integer_t pc )
```

Construct an CompiledFunction.

Parameters

argc	The count of arguments that this function expects.
pc	The bytecode address of the start of the function.

5.30.3 Member Function Documentation

5.30.3.1 __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.

5.30.3.2 __asCode()

```
string ComputedExpression::__asCode ( ) const  [virtual], [inherited]
```

Output the contents of the ComputedExpression as a string similar to how it would be represented as code.

Returns

A code-string representation of the computed expression.

Reimplemented in Tang::ComputedExpressionString.

5.30.3.3 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
             const GarbageCollected & index,
             const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.30.3.4 __boolean()

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

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.

5.30.3.5 __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.

5.30.3.6 __equal()

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

Perform an equality test.

Parameters

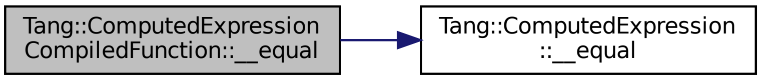
rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression CompiledFunction::__equal
Tang::ComputedExpression ::__equal

5.30.3.7 __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.

5.30.3.8 __getIterator()

```
GarbageCollected ComputedExpression::__getIterator (
            const GarbageCollected & collection ) const  [virtual], [inherited]
```

Get an iterator for the expression.

Parameters

collection	The GarbageCollected value that will serve as the collection through which to iterate.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.30.3.9 __index()

```
GarbageCollected ComputedExpression::__index (
            const GarbageCollected & index ) const  [virtual], [inherited]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.30.3.10 __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.

5.30.3.11 __iteratorNext()

```
GarbageCollected ComputedExpression::__iteratorNext (
             size_t index = 0 ) const  [virtual], [inherited]
```

Get the next iterative value.

Parameters

index	The desired index value.

Reimplemented in Tang::ComputedExpressionString, Tang::ComputedExpressionIterator, and Tang::ComputedExpressionArray.

5.30.3.12 __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.

5.30.3.13 __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.

5.30.3.14 __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.

5.30.3.15 __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.

5.30.3.16 __not()

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

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.

5.30.3.17 __slice()

```
GarbageCollected ComputedExpression::__slice (
            const GarbageCollected & begin,
            const GarbageCollected & end,
            const GarbageCollected & skip ) const  [virtual], [inherited]
```

Perform a slice operation.

Convention will follow Python semantics, in which a slice will start at the provided index position, and go up to but not including the end index. The slice will default to an index increment of 1, but can be defined as another integer value.

Parameters

begin	The begin index expression provided by the script.
end	The end index expression provided by the script.
skip	The skip index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.30.3.18 __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::ComputedExpressionIteratorEnd, Tang::ComputedExpressionInteger, Tang::ComputedExpressionFloat, Tang::ComputedExpressionError, and Tang::ComputedExpressionArray.

5.30.3.19 __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.

5.30.3.20 dump()

```
string ComputedExpressionCompiledFunction::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.

5.30.3.21 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.

5.30.3.22 is_equal() [2/6]

```
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.

5.30.3.23 is_equal() [3/6]

```
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.

5.30.3.24 is_equal() [4/6]

```
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.

5.30.3.25 is_equal() [5/6]

```
virtual bool Tang::ComputedExpression::is_equal (
             const Tang::float_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.

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

5.30.3.26 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
             const Tang::integer_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.

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

5.30.3.27 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.30.3.28 makeCopy()

```
GarbageCollected ComputedExpressionCompiledFunction::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:

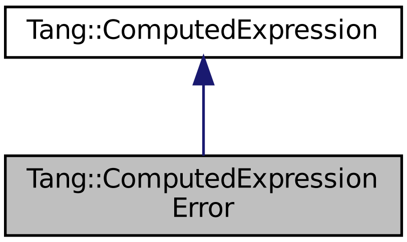
- include/computedExpressionCompiledFunction.hpp
- src/computedExpressionCompiledFunction.cpp

5.31 Tang::ComputedExpressionError Class Reference

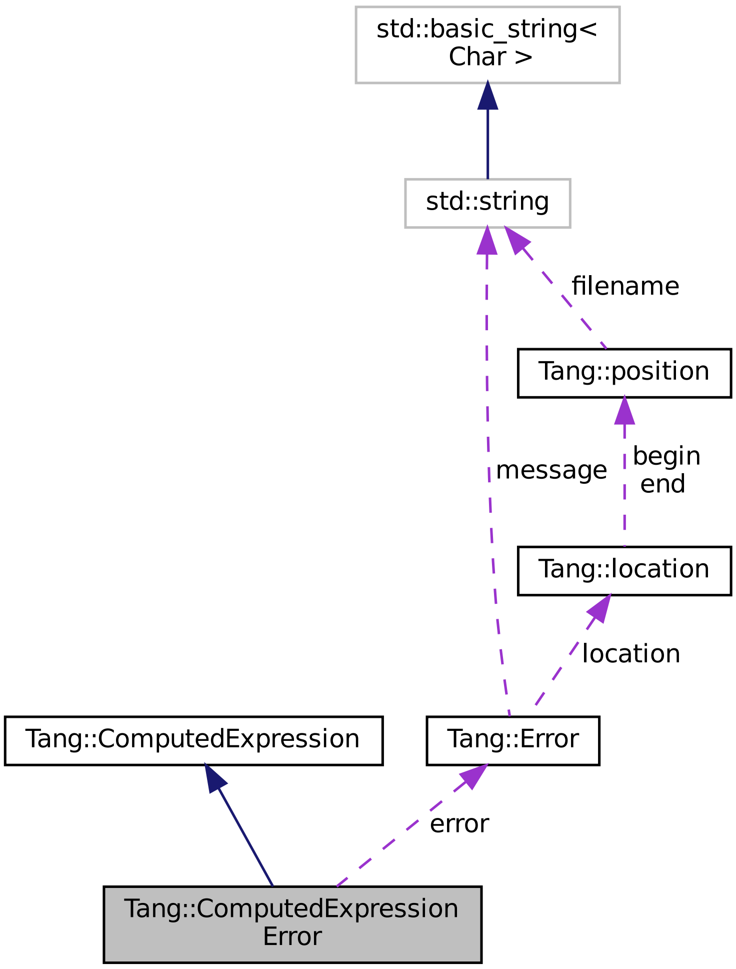
Represents a Runtime Error.

```
#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 equality 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 std::string __asCode () const

  Output the contents of the ComputedExpression as a string similar to how it would be represented as code.
- virtual bool isCopyNeeded () const

  Determine whether or not a copy is needed.
- virtual bool is_equal (const Tang::integer_t &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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.
- virtual GarbageCollected __assign_index (const GarbageCollected &index, const GarbageCollected &value)

  Perform an index assignment to the supplied value.
- virtual GarbageCollected __index (const GarbageCollected &index) const

  Perform an index operation.
- virtual GarbageCollected __slice (const GarbageCollected &begin, const GarbageCollected &end, const GarbageCollected &skip) const

  Perform a slice operation.
- virtual GarbageCollected __getIterator (const GarbageCollected &collection) const

  Get an iterator for the expression.
- virtual GarbageCollected __iteratorNext (size_t index=0) const

  Get the next iterative value.

Private Attributes

- Tang::Error error

  The Error object.

5.31.1 Detailed Description

Represents a Runtime Error.

5.31.2 Constructor & Destructor Documentation

5.31.2.1 ComputedExpressionError()

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

Construct a Runtime Error.

Parameters

error	The Tang::Error object.

5.31.3 Member Function Documentation

5.31.3.1 __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.

5.31.3.2 __asCode()

```
string ComputedExpression::__asCode ( ) const  [virtual], [inherited]
```

Output the contents of the ComputedExpression as a string similar to how it would be represented as code.

Returns

A code-string representation of the computed expression.

Reimplemented in Tang::ComputedExpressionString.

5.31.3.3 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.31.3.4 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.31.3.5 __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.

5.31.3.6 __equal()

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

Perform an equality test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.31.3.7 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.31.3.8 __getIterator()

```
GarbageCollected ComputedExpression::__getIterator (
            const GarbageCollected & collection ) const  [virtual], [inherited]
```

Get an iterator for the expression.

Parameters

collection	The GarbageCollected value that will serve as the collection through which to iterate.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.31.3.9 __index()

```
GarbageCollected ComputedExpression::__index (
            const GarbageCollected & index ) const [virtual], [inherited]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.31.3.10 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.31.3.11 __iteratorNext()

```
GarbageCollected ComputedExpression::__iteratorNext (
            size_t index = 0 ) const [virtual], [inherited]
```

Get the next iterative value.

Parameters

index	The desired index value.

Reimplemented in Tang::ComputedExpressionString, Tang::ComputedExpressionIterator, and Tang::ComputedExpressionArray.

5.31.3.12 __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.

5.31.3.13 __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.

5.31.3.14 __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.

5.31.3.15 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.31.3.16 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.31.3.17 __slice()

```
GarbageCollected ComputedExpression::__slice (
             const GarbageCollected & begin,
             const GarbageCollected & end,
             const GarbageCollected & skip ) const  [virtual], [inherited]
```

Perform a slice operation.

Convention will follow Python semantics, in which a slice will start at the provided index position, and go up to but not including the end index. The slice will default to an index increment of 1, but can be defined as another integer value.

Parameters

begin	The begin index expression provided by the script.
end	The end index expression provided by the script.
skip	The skip index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.31.3.18 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.31.3.19 __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.

5.31.3.20 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.

5.31.3.21 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.

5.31.3.22 is_equal() [2/6]

```
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.

5.31.3.23 is_equal() [3/6]

```
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.

5.31.3.24 is_equal() [4/6]

```
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.

5.31.3.25 is_equal() [5/6]

```
virtual bool Tang::ComputedExpression::is_equal (
             const Tang::float_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.

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

5.31.3.26 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
             const Tang::integer_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.

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

5.31.3.27 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.31.3.28 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:

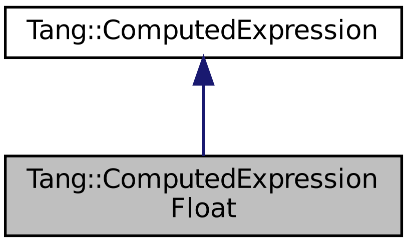
- include/computedExpressionError.hpp
- src/computedExpressionError.cpp

5.32 Tang::ComputedExpressionFloat Class Reference

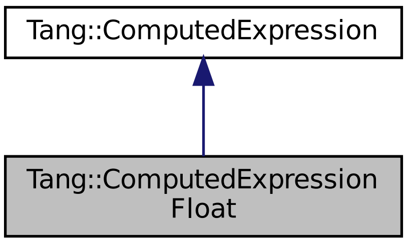
Represents a Float that is the result of a computation.

`#include <computedExpressionFloat.hpp>`

Inheritance diagram for Tang::ComputedExpressionFloat:



Collaboration diagram for Tang::ComputedExpressionFloat:



Public Member Functions

- ComputedExpressionFloat (Tang::float_t 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 Tang::integer_t &val) const override

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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 equality 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.
- Tang::float_t getValue () const

  Helper function to get the value associated with this expression.
- virtual std::string __asCode () const

  Output the contents of the ComputedExpression as a string similar to how it would be represented as code.
- virtual bool isCopyNeeded () const

  Determine whether or not a copy is needed.
- 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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)

Perform an index assignment to the supplied value.

- virtual GarbageCollected __modulo (const GarbageCollected &rhs) const

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

  Perform an index operation.
- virtual GarbageCollected __slice (const GarbageCollected &begin, const GarbageCollected &end, const GarbageCollected &skip) const

  Perform a slice operation.
- virtual GarbageCollected __getIterator (const GarbageCollected &collection) const

  Get an iterator for the expression.
- virtual GarbageCollected __iteratorNext (size_t index=0) const

  Get the next iterative value.

Private Attributes

- Tang::float_t val

  The float value.

5.32.1 Detailed Description

Represents a Float that is the result of a computation.

5.32.2 Constructor & Destructor Documentation

5.32.2.1 ComputedExpressionFloat()

```
ComputedExpressionFloat::ComputedExpressionFloat (
            Tang::float_t val )
```

Construct a Float result.

Parameters

val	The float value.

5.32.3 Member Function Documentation

5.32.3.1 __add()

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

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

Parameters

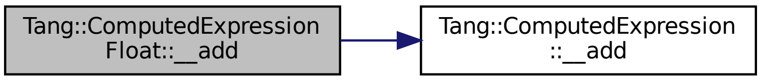
rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.32.3.2 __asCode()

```
string ComputedExpression::__asCode ( ) const  [virtual], [inherited]
```

Output the contents of the ComputedExpression as a string similar to how it would be represented as code.

Returns

A code-string representation of the computed expression.

Reimplemented in Tang::ComputedExpressionString.

5.32.3.3 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.32.3.4 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.32.3.5 __divide()

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

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

Parameters

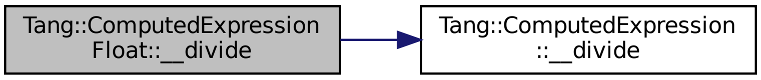
rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Float::__divide
Tang::ComputedExpression ::__divide

5.32.3.6 __equal()

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

Perform an equality test.

Parameters

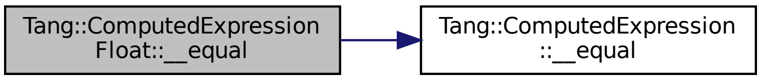
rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Float::__equal
Tang::ComputedExpression ::__equal

5.32.3.7 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.32.3.8 __getIterator()

```
GarbageCollected ComputedExpression::__getIterator (
            const GarbageCollected & collection ) const  [virtual], [inherited]
```

Get an iterator for the expression.

Parameters

collection	The GarbageCollected value that will serve as the collection through which to iterate.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.32.3.9 __index()

```
GarbageCollected ComputedExpression::__index (
             const GarbageCollected & index ) const  [virtual], [inherited]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.32.3.10 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.32.3.11 __iteratorNext()

```
GarbageCollected ComputedExpression::__iteratorNext (
             size_t index = 0 ) const  [virtual], [inherited]
```

Get the next iterative value.

Parameters

index	The desired index value.

Reimplemented in Tang::ComputedExpressionString, Tang::ComputedExpressionIterator, and Tang::ComputedExpressionArray.

5.32.3.12 __lessThan()

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

Compute the "less than" comparison.

Parameters

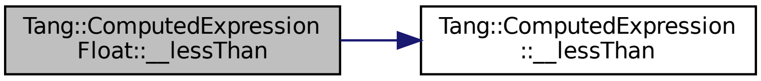
rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Float::__lessThan
Tang::ComputedExpression ::__lessThan

5.32.3.13 __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.

5.32.3.14 __multiply()

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

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

Parameters

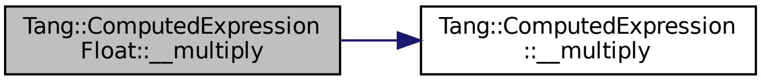
rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Float::__multiply
Tang::ComputedExpression ::__multiply

5.32.3.15 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.32.3.16 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.32.3.17 __slice()

```
GarbageCollected ComputedExpression::__slice (
            const GarbageCollected & begin,
            const GarbageCollected & end,
            const GarbageCollected & skip ) const  [virtual], [inherited]
```

Perform a slice operation.

Convention will follow Python semantics, in which a slice will start at the provided index position, and go up to but not including the end index. The slice will default to an index increment of 1, but can be defined as another integer value.

Parameters

begin	The begin index expression provided by the script.
end	The end index expression provided by the script.
skip	The skip index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.32.3.18 __string()

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

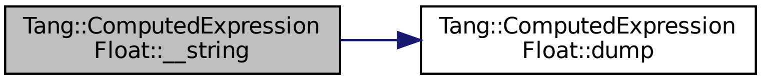
Perform a type cast to string.

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

5.32.3.19 __subtract()

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

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

Parameters

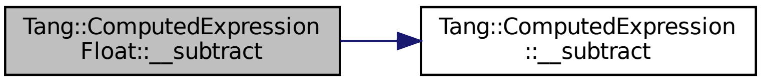
rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Float::__subtract
Tang::ComputedExpression ::__subtract

5.32.3.20 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.

5.32.3.21 getValue()

```
Tang::float_t ComputedExpressionFloat::getValue ( ) const
```

Helper function to get the value associated with this expression.

Returns

The value associated with this expression.

5.32.3.22 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.

5.32.3.23 is_equal() [2/6]

```
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.

5.32.3.24 is_equal() [3/6]

```
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.

5.32.3.25 is_equal() [4/6]

```
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.

5.32.3.26 is_equal() [5/6]

```
bool ComputedExpressionFloat::is_equal (
            const Tang::float_t & 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.

5.32.3.27 is_equal() [6/6]

```
bool ComputedExpressionFloat::is_equal (
            const Tang::integer_t & 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.

5.32.3.28 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.32.3.29 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:

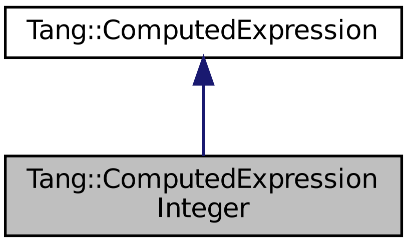
- include/computedExpressionFloat.hpp
- src/computedExpressionFloat.cpp

5.33 Tang::ComputedExpressionInteger Class Reference

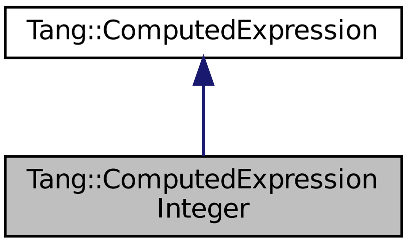
Represents an Integer that is the result of a computation.

`#include <computedExpressionInteger.hpp>`

Inheritance diagram for Tang::ComputedExpressionInteger:



Collaboration diagram for Tang::ComputedExpressionInteger:



Public Member Functions

- ComputedExpressionInteger (Tang::integer_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 Tang::integer_t &val) const override

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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 equality 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.
- Tang::integer_t getValue () const

  Helper function to get the value associated with this expression.
- virtual std::string __asCode () const

  Output the contents of the ComputedExpression as a string similar to how it would be represented as code.
- virtual bool isCopyNeeded () const

  Determine whether or not a copy is needed.
- 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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)

  Perform an index assignment to the supplied value.
- virtual GarbageCollected __index (const GarbageCollected &index) const

  Perform an index operation.
- virtual GarbageCollected __slice (const GarbageCollected &begin, const GarbageCollected &end, const GarbageCollected &skip) const

  Perform a slice operation.
- virtual GarbageCollected __getIterator (const GarbageCollected &collection) const

  Get an iterator for the expression.
- virtual GarbageCollected __iteratorNext (size_t index=0) const

  Get the next iterative value.

Private Attributes

- Tang::integer_t val

  The integer value.

5.33.1 Detailed Description

Represents an Integer that is the result of a computation.

5.33.2 Constructor & Destructor Documentation

5.33.2.1 ComputedExpressionInteger()

```
ComputedExpressionInteger::ComputedExpressionInteger (
            Tang::integer_t val )
```

Construct an Integer result.

Parameters

val	The integer value.

5.33.3 Member Function Documentation

5.33.3.1 __add()

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

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

Parameters

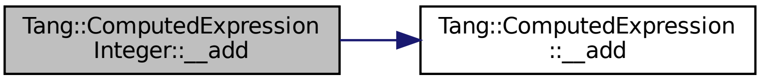
rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Integer::__add
Tang::ComputedExpression ::__add

5.33.3.2 __asCode()

```
string ComputedExpression::__asCode ( ) const  [virtual], [inherited]
```

Output the contents of the ComputedExpression as a string similar to how it would be represented as code.

Returns

A code-string representation of the computed expression.

Reimplemented in Tang::ComputedExpressionString.

5.33.3.3 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.33.3.4 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.33.3.5 __divide()

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

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

Parameters

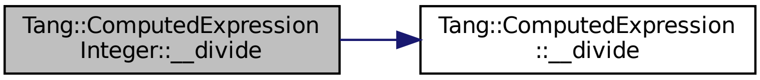
rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Integer::__divide
Tang::ComputedExpression ::__divide

5.33.3.6 __equal()

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

Perform an equality test.

Parameters

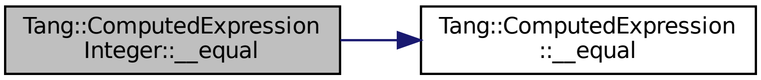
rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Integer::__equal
Tang::ComputedExpression ::__equal

5.33.3.7 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.33.3.8 __getIterator()

```
GarbageCollected ComputedExpression::__getIterator (
            const GarbageCollected & collection ) const  [virtual], [inherited]
```

Get an iterator for the expression.

Parameters

collection	The GarbageCollected value that will serve as the collection through which to iterate.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.33.3.9 __index()

```
GarbageCollected ComputedExpression::__index (
            const GarbageCollected & index ) const  [virtual], [inherited]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.33.3.10 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.33.3.11 __iteratorNext()

```
GarbageCollected ComputedExpression::__iteratorNext (
            size_t index = 0 ) const  [virtual], [inherited]
```

Get the next iterative value.

Parameters

index	The desired index value.

Reimplemented in Tang::ComputedExpressionString, Tang::ComputedExpressionIterator, and Tang::ComputedExpressionArray.

5.33.3.12 __lessThan()

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

Compute the "less than" comparison.

Parameters

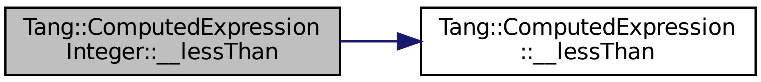
rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Integer::__lessThan
Tang::ComputedExpression ::__lessThan

5.33.3.13 __modulo()

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

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

Parameters

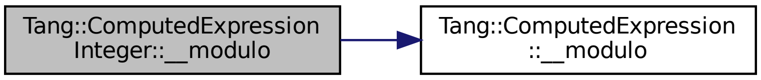
rhs	The GarbageCollected value to modulo this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Integer::__modulo
Tang::ComputedExpression ::__modulo

5.33.3.14 __multiply()

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

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

Parameters

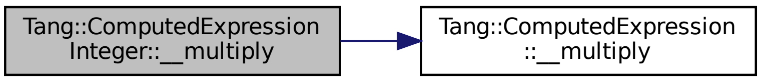
rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Integer::__multiply
Tang::ComputedExpression ::__multiply

5.33.3.15 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.33.3.16 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.33.3.17 __slice()

```
GarbageCollected ComputedExpression::__slice (
             const GarbageCollected & begin,
             const GarbageCollected & end,
             const GarbageCollected & skip ) const  [virtual], [inherited]
```

Perform a slice operation.

Convention will follow Python semantics, in which a slice will start at the provided index position, and go up to but not including the end index. The slice will default to an index increment of 1, but can be defined as another integer value.

Parameters

begin	The begin index expression provided by the script.
end	The end index expression provided by the script.
skip	The skip index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.33.3.18 __string()

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

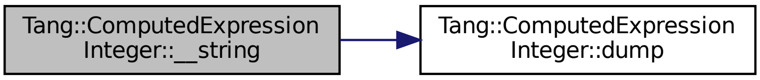
Perform a type cast to string.

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

5.33.3.19 __subtract()

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

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

Parameters

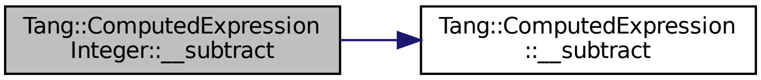
rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression Integer::__subtract
Tang::ComputedExpression ::__subtract

5.33.3.20 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.

5.33.3.21 getValue()

```
Tang::integer_t ComputedExpressionInteger::getValue ( ) const
```

Helper function to get the value associated with this expression.

Returns

The value associated with this expression.

5.33.3.22 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.

5.33.3.23 is_equal() [2/6]

```
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.

5.33.3.24 is_equal() [3/6]

```
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.

5.33.3.25 is_equal() [4/6]

```
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.

5.33.3.26 is_equal() [5/6]

```
bool ComputedExpressionInteger::is_equal (
            const Tang::float_t & 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.

5.33.3.27 is_equal() [6/6]

```
bool ComputedExpressionInteger::is_equal (
            const Tang::integer_t & 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.

5.33.3.28 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.33.3.29 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:

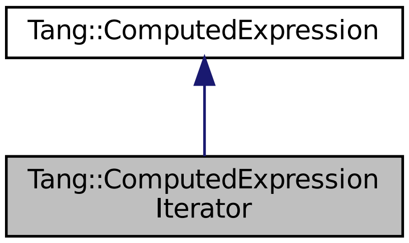
- include/computedExpressionInteger.hpp
- src/computedExpressionInteger.cpp

5.34 Tang::ComputedExpressionIterator Class Reference

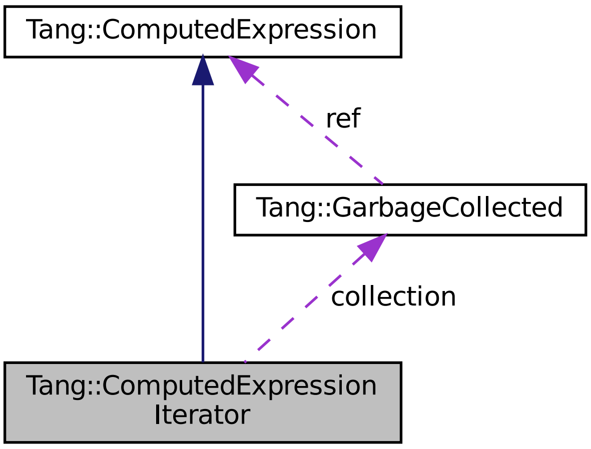
Represents an Iterator that is the result of a computation.

```
#include <computedExpressionIterator.hpp>
```

Inheritance diagram for Tang::ComputedExpressionIterator:



Collaboration diagram for Tang::ComputedExpressionIterator:



Public Member Functions

- ComputedExpressionIterator (Tang::GarbageCollected collection)

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

  Output the contents of the ComputedExpression as a string.
- virtual GarbageCollected __iteratorNext (size_t index) const override

  Get the next iterative value.
- virtual std::string __asCode () const

  Output the contents of the ComputedExpression as a string similar to how it would be represented as code.
- virtual bool isCopyNeeded () const

  Determine whether or not a copy is needed.
- virtual GarbageCollected makeCopy () const

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

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)

  Perform an index assignment to the supplied 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 equality test.
- virtual GarbageCollected __index (const GarbageCollected &index) const

  Perform an index operation.
- virtual GarbageCollected __slice (const GarbageCollected &begin, const GarbageCollected &end, const GarbageCollected &skip) const

  Perform a slice operation.
- virtual GarbageCollected __getIterator (const GarbageCollected &collection) const

  Get an iterator for the expression.
- 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.

Private Attributes

- Tang::GarbageCollected collection

  The target collection.
- size_t index

  The next index.

5.34.1 Detailed Description

Represents an Iterator that is the result of a computation.

5.34.2 Constructor & Destructor Documentation

5.34.2.1 ComputedExpressionIterator()

```
ComputedExpressionIterator::ComputedExpressionIterator (
            Tang::GarbageCollected collection )
```

Construct an Iterator result.

Parameters

collection	The collection through which the iterator processes

5.34.3 Member Function Documentation

5.34.3.1 __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.

5.34.3.2 __asCode()

```
string ComputedExpression::__asCode ( ) const  [virtual], [inherited]
```

Output the contents of the ComputedExpression as a string similar to how it would be represented as code.

Returns

A code-string representation of the computed expression.

Reimplemented in Tang::ComputedExpressionString.

5.34.3.3 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.34.3.4 __boolean()

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

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.

5.34.3.5 __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.

5.34.3.6 __equal()

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

Perform an equality 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, Tang::ComputedExpressionCompiledFunction, and Tang::ComputedExpressionBoolean.

5.34.3.7 __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.

5.34.3.8 __getIterator()

```
GarbageCollected ComputedExpression::__getIterator (
            const GarbageCollected & collection ) const  [virtual], [inherited]
```

Get an iterator for the expression.

Parameters

collection	The GarbageCollected value that will serve as the collection through which to iterate.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.34.3.9 __index()

```
GarbageCollected ComputedExpression::__index (
            const GarbageCollected & index ) const [virtual], [inherited]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.34.3.10 __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.

5.34.3.11 __iteratorNext()

```
GarbageCollected ComputedExpressionIterator::__iteratorNext (
            size_t index ) const [override], [virtual]
```

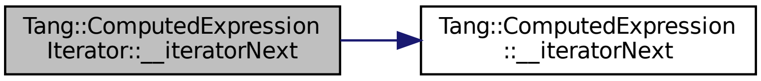
Get the next iterative value.

Parameters

index	The desired index value.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.34.3.12 __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.

5.34.3.13 __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.

5.34.3.14 __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.

5.34.3.15 __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.

5.34.3.16 __not()

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

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.

5.34.3.17 __slice()

```
GarbageCollected ComputedExpression::__slice (
            const GarbageCollected & begin,
            const GarbageCollected & end,
            const GarbageCollected & skip ) const  [virtual], [inherited]
```

Perform a slice operation.

Convention will follow Python semantics, in which a slice will start at the provided index position, and go up to but not including the end index. The slice will default to an index increment of 1, but can be defined as another integer value.

Parameters

begin	The begin index expression provided by the script.
end	The end index expression provided by the script.
skip	The skip index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.34.3.18 __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::ComputedExpressionIteratorEnd, Tang::ComputedExpressionInteger, Tang::ComputedExpressionFloat, Tang::ComputedExpressionError, and Tang::ComputedExpressionArray.

5.34.3.19 __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.

5.34.3.20 dump()

```
string ComputedExpressionIterator::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.

5.34.3.21 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.

5.34.3.22 is_equal() [2/6]

```
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.

5.34.3.23 is_equal() [3/6]

```
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.

5.34.3.24 is_equal() [4/6]

```
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.

5.34.3.25 is_equal() [5/6]

```
virtual bool Tang::ComputedExpression::is_equal (
             const Tang::float_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.

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

5.34.3.26 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
             const Tang::integer_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.

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

5.34.3.27 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.34.3.28 makeCopy()

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

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, Tang::ComputedExpressionCompiledFunction, Tang::ComputedExpressionBoolean, and Tang::ComputedExpressionArray.

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

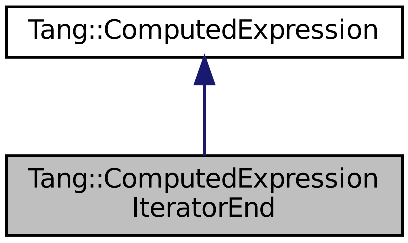
- include/computedExpressionIterator.hpp
- src/computedExpressionIterator.cpp

5.35 Tang::ComputedExpressionIteratorEnd Class Reference

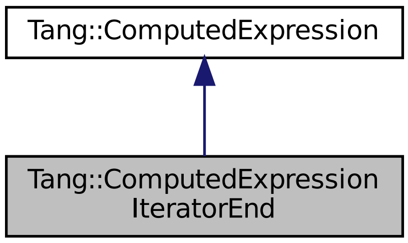
Represents that a collection has no more values through which to iterate.

`#include <computedExpressionIteratorEnd.hpp>`

Inheritance diagram for Tang::ComputedExpressionIteratorEnd:



Collaboration diagram for Tang::ComputedExpressionIteratorEnd:



Public Member Functions

- ComputedExpressionIteratorEnd ()
  Construct an IteratorEnd result.
- virtual std::string dump () const override
  Output the contents of the ComputedExpression as a string.
- virtual GarbageCollected __string () const override
  Perform a type cast to string.
- virtual std::string __asCode () const
  Output the contents of the ComputedExpression as a string similar to how it would be represented as code.
- virtual bool isCopyNeeded () const
  Determine whether or not a copy is needed.
- virtual GarbageCollected makeCopy () const
  Make a copy of the ComputedExpression (recursively, if appropriate).
- virtual bool is_equal (const Tang::integer_t &val) const
  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)
  Perform an index assignment to the supplied 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 equality test.
- virtual GarbageCollected __index (const GarbageCollected &index) const
  Perform an index operation.
- virtual GarbageCollected __slice (const GarbageCollected &begin, const GarbageCollected &end, const GarbageCollected &skip) const
  Perform a slice operation.

- virtual GarbageCollected __getIterator (const GarbageCollected &collection) const

  Get an iterator for the expression.
- virtual GarbageCollected __iteratorNext (size_t index=0) const

  Get the next iterative value.
- virtual GarbageCollected __integer () const

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

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

  Perform a type cast to boolean.

5.35.1 Detailed Description

Represents that a collection has no more values through which to iterate.

5.35.2 Member Function Documentation

5.35.2.1 __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.

5.35.2.2 __asCode()

```
string ComputedExpression::__asCode ( ) const  [virtual], [inherited]
```

Output the contents of the ComputedExpression as a string similar to how it would be represented as code.

Returns

A code-string representation of the computed expression.

Reimplemented in Tang::ComputedExpressionString.

5.35.2.3 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.35.2.4 __boolean()

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

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.

5.35.2.5 __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.

5.35.2.6 __equal()

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

Perform an equality 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, Tang::ComputedExpressionCompiledFunction, and Tang::ComputedExpressionBoolean.

5.35.2.7 __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.

5.35.2.8 __getIterator()

```
GarbageCollected ComputedExpression::__getIterator (
            const GarbageCollected & collection ) const  [virtual], [inherited]
```

Get an iterator for the expression.

Parameters

collection	The GarbageCollected value that will serve as the collection through which to iterate.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.35.2.9 __index()

```
GarbageCollected ComputedExpression::__index (
            const GarbageCollected & index ) const  [virtual], [inherited]
```

Perform an index operation.

Parameters

index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.35.2.10 __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.

5.35.2.11 __iteratorNext()

```
GarbageCollected ComputedExpression::__iteratorNext (
            size_t index = 0 ) const  [virtual], [inherited]
```

Get the next iterative value.

Parameters

index	The desired index value.

Reimplemented in Tang::ComputedExpressionString, Tang::ComputedExpressionIterator, and Tang::ComputedExpressionArray.

5.35.2.12 __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.

5.35.2.13 __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.

5.35.2.14 __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.

5.35.2.15 __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.

5.35.2.16 __not()

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

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.

5.35.2.17 __slice()

```
GarbageCollected ComputedExpression::__slice (
            const GarbageCollected & begin,
            const GarbageCollected & end,
            const GarbageCollected & skip ) const  [virtual], [inherited]
```

Perform a slice operation.

Convention will follow Python semantics, in which a slice will start at the provided index position, and go up to but not including the end index. The slice will default to an index increment of 1, but can be defined as another integer value.

Parameters

begin	The begin index expression provided by the script.
end	The end index expression provided by the script.
skip	The skip index expression provided by the script.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionString, and Tang::ComputedExpressionArray.

5.35.2.18 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.35.2.19 __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.

5.35.2.20 dump()

```
string ComputedExpressionIteratorEnd::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.

5.35.2.21 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.

5.35.2.22 is_equal() [2/6]

```
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.

5.35.2.23 is_equal() [3/6]

```
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.

5.35.2.24 is_equal() [4/6]

```
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.

5.35.2.25 is_equal() [5/6]

```
virtual bool Tang::ComputedExpression::is_equal (
             const Tang::float_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.

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

5.35.2.26 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
             const Tang::integer_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.

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

5.35.2.27 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.35.2.28 makeCopy()

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

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, Tang::ComputedExpressionCompiledFunction, Tang::ComputedExpressionBoolean, and Tang::ComputedExpressionArray.

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

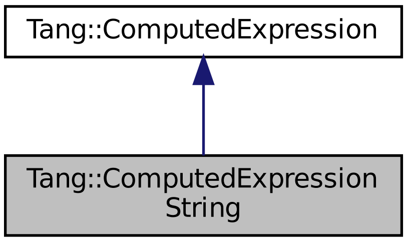
- include/computedExpressionIteratorEnd.hpp
- src/computedExpressionIteratorEnd.cpp

5.36 Tang::ComputedExpressionString Class Reference

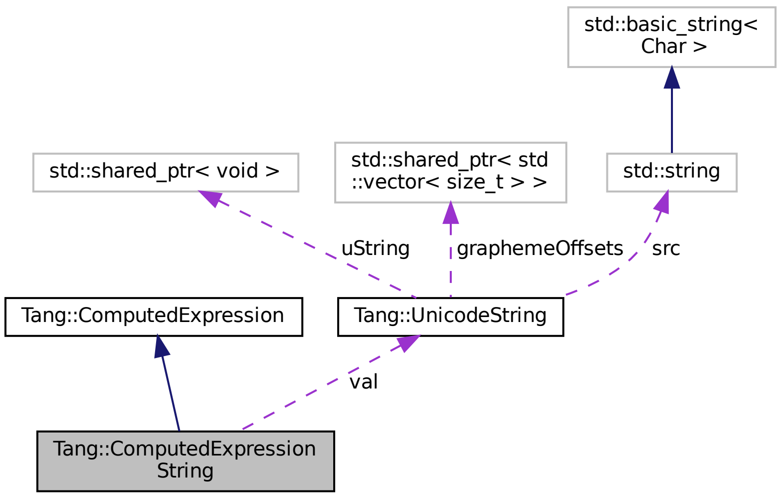
Represents a String that is the result of a computation.

```
#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.
- virtual std::string __asCode () const override

  Output the contents of the ComputedExpression as a string similar to how it would be represented as code.
- 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 __index (const GarbageCollected &index) const override

  Perform an index operation.
- virtual GarbageCollected __slice (const GarbageCollected &begin, const GarbageCollected &end, const GarbageCollected &skip) const override

  Perform a slice operation.
- virtual GarbageCollected __getIterator (const GarbageCollected &collection) const override

  Get an iterator for the expression.
- virtual GarbageCollected __iteratorNext (size_t index) const override

  Get the next iterative 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 equality 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 isCopyNeeded () const

  Determine whether or not a copy is needed.
- virtual bool is_equal (const Tang::integer_t &val) const

  Check whether or not the computed expression is equal to another value.
- virtual bool is_equal (const Tang::float_t &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 __assign_index (const GarbageCollected &index, const GarbageCollected &value)

  Perform an index assignment to the supplied value.
- virtual GarbageCollected __subtract (const GarbageCollected &rhs) const

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

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

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

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

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

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

  Perform a type cast to float.

Private Attributes

- UnicodeString val

  The string value.

5.36.1 Detailed Description

Represents a String that is the result of a computation.

5.36.2 Constructor & Destructor Documentation

5.36.2.1 ComputedExpressionString()

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

Construct a String result.

Parameters

val	The string value.

5.36.3 Member Function Documentation

5.36.3.1 __add()

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

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

Parameters

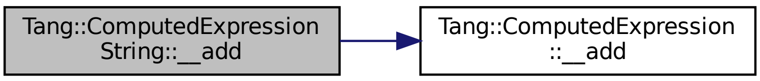
rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.36.3.2 __asCode()

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

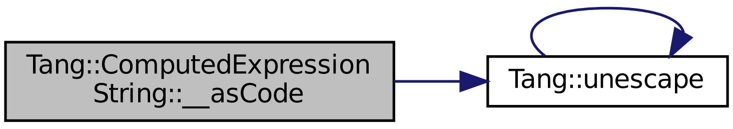
Output the contents of the ComputedExpression as a string similar to how it would be represented as code.

Returns

A code-string representation of the computed expression.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.36.3.3 __assign_index()

```
GarbageCollected ComputedExpression::__assign_index (
            const GarbageCollected & index,
            const GarbageCollected & value )  [virtual], [inherited]
```

Perform an index assignment to the supplied value.

Parameters

index	The index to which the value should be applied.
value	The value to store.

Returns

The result of the operation.

Reimplemented in Tang::ComputedExpressionArray.

5.36.3.4 __boolean()

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

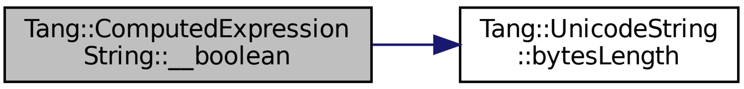
Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.36.3.5 __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.

5.36.3.6 __equal()

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

Perform an equality test.

Parameters

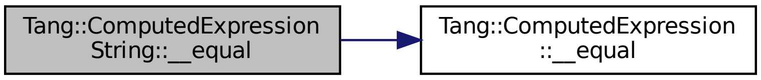
rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression String::__equal
Tang::ComputedExpression ::__equal

5.36.3.7 __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.

5.36.3.8 __getIterator()

```
GarbageCollected ComputedExpressionString::__getIterator (
            const GarbageCollected & collection ) const  [override], [virtual]
```

Get an iterator for the expression.

Parameters

collection	The GarbageCollected value that will serve as the collection through which to iterate.

Reimplemented from Tang::ComputedExpression.

5.36.3.9 __index()

```
GarbageCollected ComputedExpressionString::__index (
            const GarbageCollected & index ) const  [override], [virtual]
```

Perform an index operation.

Parameters

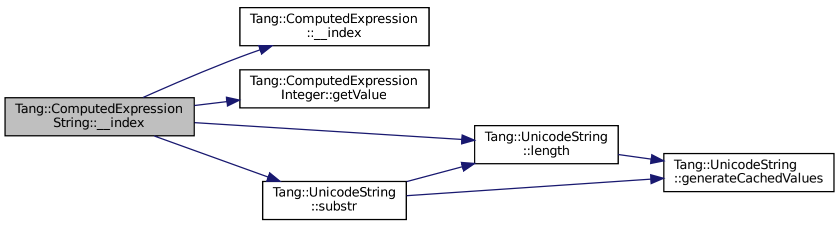
index	The index expression provided by the script.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression ::__index
Tang::ComputedExpression Integer::getValue
Tang::ComputedExpression String::__index
Tang::UnicodeString ::length
Tang::UnicodeString ::generateCachedValues
Tang::UnicodeString ::substr

5.36.3.10 __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.

5.36.3.11 __iteratorNext()

```
GarbageCollected ComputedExpressionString::__iteratorNext (
            size_t index ) const  [override], [virtual]
```

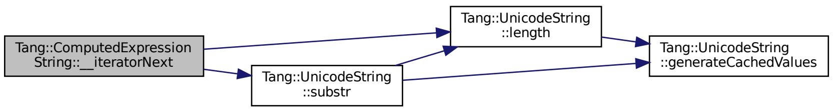
Get the next iterative value.

Parameters

index	The desired index value.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression String::__iteratorNext
Tang::UnicodeString ::length
Tang::UnicodeString ::substr
Tang::UnicodeString ::generateCachedValues

5.36.3.12 __lessThan()

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

Compute the "less than" comparison.

Parameters

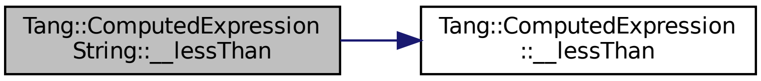
rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression String::__lessThan
Tang::ComputedExpression ::__lessThan

5.36.3.13 __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.

5.36.3.14 __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.

5.36.3.15 __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.

5.36.3.16 __not()

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

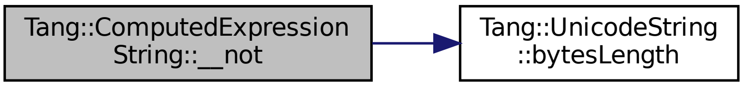
Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression String::__not
Tang::UnicodeString ::bytesLength

5.36.3.17 __slice()

```
GarbageCollected ComputedExpressionString::__slice (
             const GarbageCollected & begin,
             const GarbageCollected & end,
             const GarbageCollected & skip ) const  [override], [virtual]
```

Perform a slice operation.

Convention will follow Python semantics, in which a slice will start at the provided index position, and go up to but not including the end index. The slice will default to an index increment of 1, but can be defined as another integer value.

Parameters

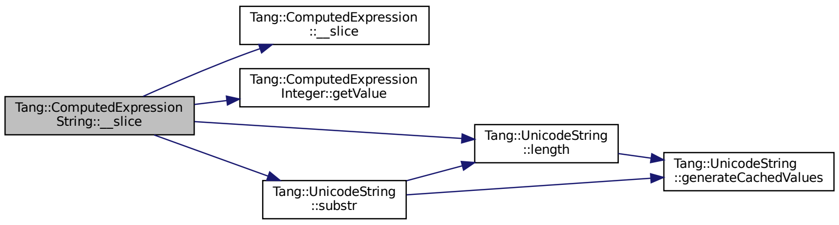
begin	The begin index expression provided by the script.
end	The end index expression provided by the script.
skip	The skip index expression provided by the script.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:


Tang::ComputedExpression ::__slice
Tang::ComputedExpression Integer::getValue
Tang::ComputedExpression String::__slice
Tang::UnicodeString ::length
Tang::UnicodeString ::generateCachedValues
Tang::UnicodeString ::substr

5.36.3.18 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.36.3.19 __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.

5.36.3.20 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.

5.36.3.21 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

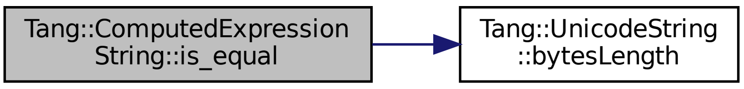
val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.36.3.22 is_equal() [2/6]

```
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.

5.36.3.23 is_equal() [3/6]

```
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.

5.36.3.24 is_equal() [4/6]

```
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.

5.36.3.25 is_equal() [5/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::float_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.

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

5.36.3.26 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const Tang::integer_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.

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

5.36.3.27 isCopyNeeded()

```
bool ComputedExpression::isCopyNeeded ( ) const  [virtual], [inherited]
```

Determine whether or not a copy is needed.

Copying is only required for ComputedExpressions which serve as containers, such as ComputedExpressionArray and ComputedExpressionObject.

Returns

Whether or not a copy is needed.

Reimplemented in Tang::ComputedExpressionArray.

5.36.3.28 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:

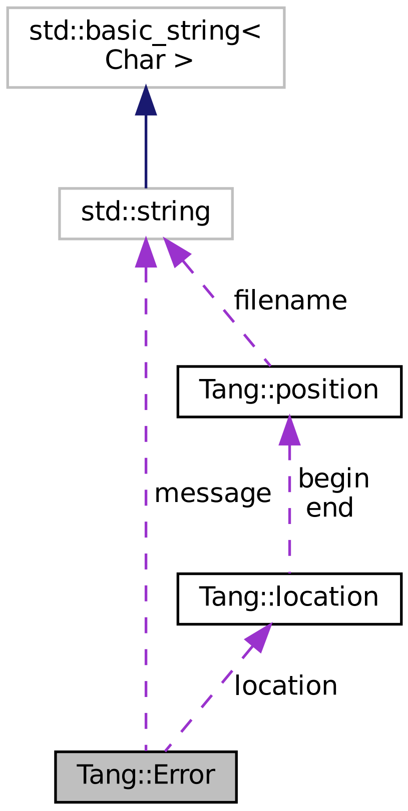
- include/computedExpressionString.hpp
- src/computedExpressionString.cpp

5.37 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.

5.37.1 Detailed Description

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

5.37.2 Constructor & Destructor Documentation

5.37.2.1 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.

5.37.2.2 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.

5.37.3 Friends And Related Function Documentation

5.37.3.1 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:

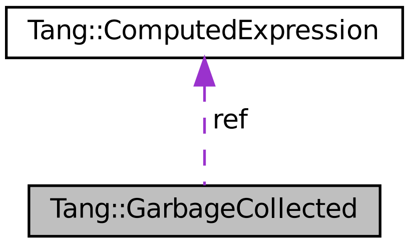
- include/error.hpp
- src/error.cpp

5.38 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.
- bool isCopyNeeded () const

  Determine whether or not a copy is needed as determined by the referenced ComputedExpression.
- GarbageCollected makeCopy () const

  Create a separate copy of the original GarbageCollected value.
- ComputedExpression ∗ operator-> () const

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

  Access the tracked object.
- bool operator== (const Tang::integer_t &val) const

  Compare the GarbageCollected tracked object with a supplied value.
- bool operator== (const Tang::float_t &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.

5.38.1 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.

5.38.2 Constructor & Destructor Documentation

5.38.2.1 GarbageCollected() `[1/3]`

```
GarbageCollected::GarbageCollected (
            const GarbageCollected & other )
```

Copy Constructor.

Parameters

The	other GarbageCollected object to copy.

5.38.2.2 GarbageCollected() `[2/3]`

```
GarbageCollected::GarbageCollected (
            GarbageCollected && other )
```

Move Constructor.

Parameters

The	other GarbageCollected object to move.

5.38.2.3 ∼GarbageCollected()

```
GarbageCollected::∼GarbageCollected ( )
```

Destructor.

Clean up the tracked object, if appropriate.

5.38.2.4 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.

5.38.3 Member Function Documentation

5.38.3.1 isCopyNeeded()

```
bool GarbageCollected::isCopyNeeded ( ) const
```

Determine whether or not a copy is needed as determined by the referenced ComputedExpression.

Returns

Whether or not a copy is needed.

5.38.3.2 make()

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

Creates a garbage-collected object of the specified type.

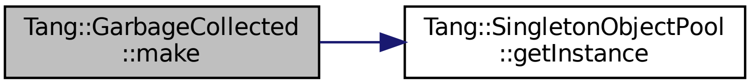
Parameters

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

5.38.3.3 makeCopy()

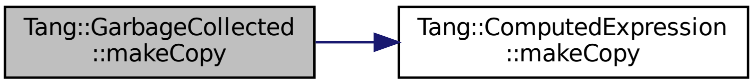
```
GarbageCollected GarbageCollected::makeCopy ( ) const
```

Create a separate copy of the original GarbageCollected value.

Returns

A GarbageCollected copy of the original value.

Here is the call graph for this function:


Tang::GarbageCollected ::makeCopy
Tang::ComputedExpression ::makeCopy

5.38.3.4 operator"!()

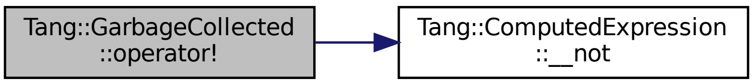
```
GarbageCollected GarbageCollected::operator!  ( ) const
```

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

5.38.3.5 operator"!=()

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

Perform a != between two GarbageCollected values.

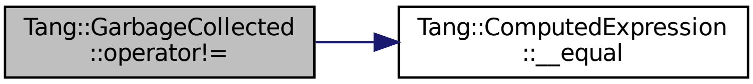
Parameters

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

5.38.3.6 operator%()

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

Perform a modulo between two GarbageCollected values.

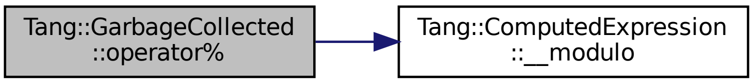
Parameters

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

5.38.3.7 operator∗() [1/2]

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

Access the tracked object.

Returns

A reference to the tracked object.

5.38.3.8 operator∗() [2/2]

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

Perform a multiplication between two GarbageCollected values.

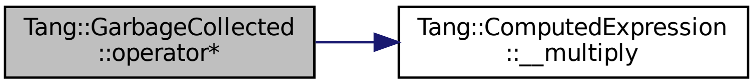
Parameters

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

5.38.3.9 operator+()

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

Perform an addition between two GarbageCollected values.

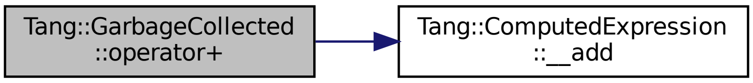
Parameters

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

5.38.3.10 operator-() [1/2]

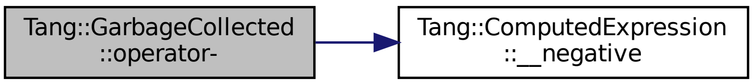
```
GarbageCollected GarbageCollected::operator- ( ) const
```

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

5.38.3.11 operator-() `[2/2]`

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

Perform a subtraction between two GarbageCollected values.

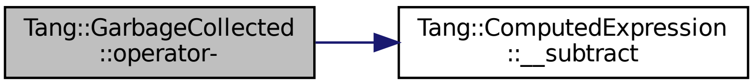
Parameters

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

5.38.3.12 operator->()

```
ComputedExpression * GarbageCollected::operator-> ( ) const
```

Access the tracked object as a pointer.

Returns

A pointer to the tracked object.

5.38.3.13 operator/()

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

Perform a division between two GarbageCollected values.

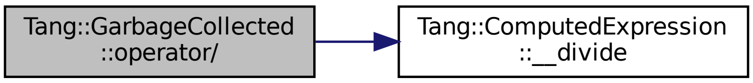
Parameters

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

5.38.3.14 operator<()

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

Perform a $<$ between two GarbageCollected values.

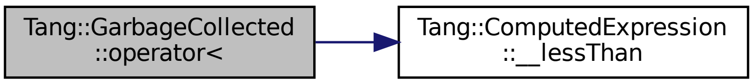
Parameters

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

5.38.3.15 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.

5.38.3.16 operator=() [1/2]

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

Copy Assignment.

Parameters

The	other GarbageCollected object.

5.38.3.17 operator=() [2/2]

```
GarbageCollected & GarbageCollected::operator= (
             GarbageCollected && other )
```

Move Assignment.

Parameters

The	other GarbageCollected object.

5.38.3.18 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.

5.38.3.19 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.

5.38.3.20 operator==() [3/8]

```
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.

5.38.3.21 operator==() [4/8]

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

Perform a == between two GarbageCollected values.

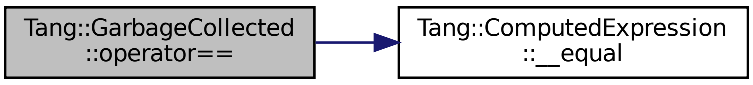
Parameters

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

5.38.3.22 operator==() [5/8]

```
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.

5.38.3.23 operator==() [6/8]

```
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.

5.38.3.24 operator==() [7/8]

```
bool GarbageCollected::operator== (
             const Tang::float_t & 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.

5.38.3.25 operator==() [8/8]

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

Compare the GarbageCollected tracked object with a supplied value.

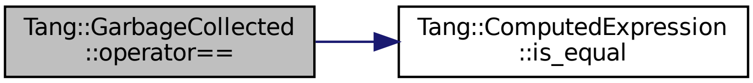
Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

Here is the call graph for this function:


Tang::GarbageCollected ::operator==
Tang::ComputedExpression ::is_equal

5.38.3.26 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.

5.38.3.27 operator>=()

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

Perform a >= between two GarbageCollected values.

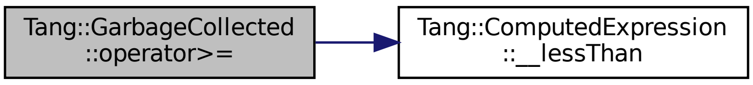
Parameters

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

5.38.4 Friends And Related Function Documentation

5.38.4.1 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:

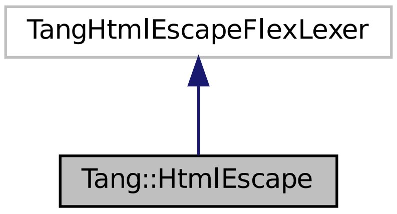
- include/garbageCollected.hpp
- src/garbageCollected.cpp

5.39 Tang::HtmlEscape Class Reference

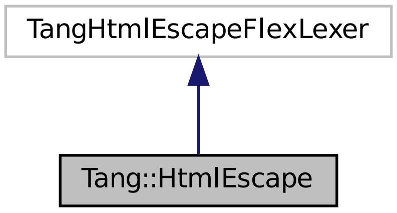
The Flex lexer class for the main Tang language.

```
#include <htmlEscape.hpp>
```

Inheritance diagram for Tang::HtmlEscape:



Collaboration diagram for Tang::HtmlEscape:



Public Member Functions

- HtmlEscape (std::istream &arg_yyin, std::ostream &arg_yyout)

  The constructor for the Scanner.
- virtual std::string get_next_token ()

  Extract the next token from the input string.

5.39.1 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.

5.39.2 Constructor & Destructor Documentation

5.39.2.1 HtmlEscape()

```
Tang::HtmlEscape::HtmlEscape (
            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)

5.39.3 Member Function Documentation

5.39.3.1 get_next_token()

```
virtual std::string Tang::HtmlEscape::get_next_token ( )  [virtual]
```

Extract the next token from the input string.

Returns

The next unescaped character.

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

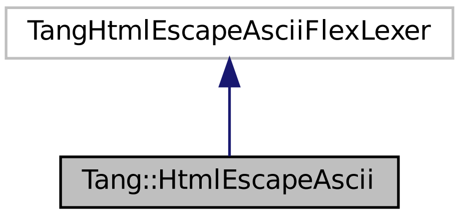
- include/htmlEscape.hpp

5.40 Tang::HtmlEscapeAscii Class Reference

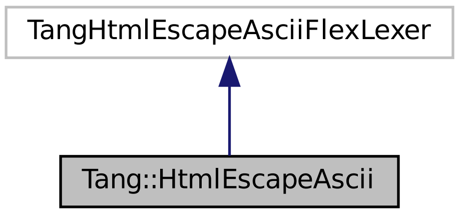
The Flex lexer class for the main Tang language.

```
#include <htmlEscapeAscii.hpp>
```

Inheritance diagram for Tang::HtmlEscapeAscii:



Collaboration diagram for Tang::HtmlEscapeAscii:



Public Member Functions

- HtmlEscapeAscii (std::istream &arg_yyin, std::ostream &arg_yyout)

  The constructor for the Scanner.
- virtual std::string get_next_token ()

  Extract the next token from the input string.

5.40.1 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.

5.40.2 Constructor & Destructor Documentation

5.40.2.1 HtmlEscapeAscii()

```
Tang::HtmlEscapeAscii::HtmlEscapeAscii (
            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)

5.40.3 Member Function Documentation

5.40.3.1 get_next_token()

```
virtual std::string Tang::HtmlEscapeAscii::get_next_token ( )  [virtual]
```

Extract the next token from the input string.

Returns

The next unescaped character.

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

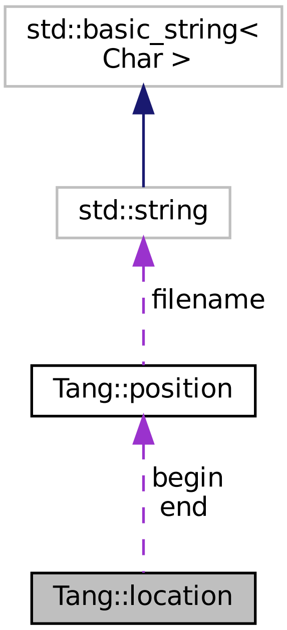
- include/htmlEscapeAscii.hpp

5.41 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.

5.41.1 Detailed Description

Two points in a source file.

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

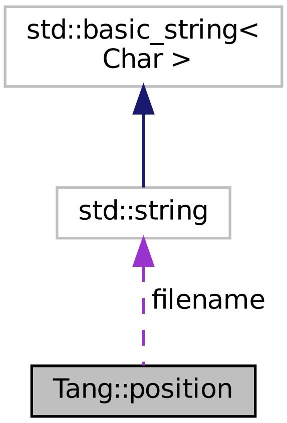
- build/generated/location.hh

5.42 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.

Static Private Member Functions

- static counter_type add_ (counter_type lhs, counter_type rhs, counter_type min)

  Compute max (min, lhs+rhs).

5.42.1 Detailed Description

A point in a source file.

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

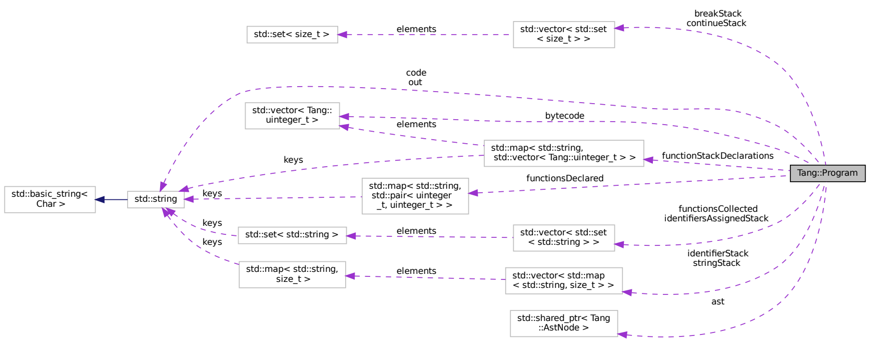
- build/generated/location.hh

5.43 Tang::Program Class Reference

Represents a compiled script or template that may be executed.

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

Collaboration diagram for Tang::Program:


std::set< size_t >
elements
std::vector< std::set < size_t > >
breakStack continueStack
code out
std::vector< Tang:: uinteger_t >
bytecode
elements
std::map< std::string, std::vector< Tang::uinteger_t > >
functionStackDeclarations
keys
Tang::Program
functionsDeclared
std::basic_string< Char >
std::string
keys
std::map< std::string, std::pair< uinteger _t, uinteger_t > >
functionsCollected identifiersAssignedStack
keys
std::set< std::string >
elements
std::vector< std::set < std::string > >
keys
identifierStack stringStack
std::map< std::string, size_t >
elements
std::vector< std::map < std::string, size_t > >
ast
std::shared_ptr< Tang ::AstNode >

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 (Tang::uinteger_t)

    Add a Tang::uinteger_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, Tang::uinteger_t jumpTarget)

    Set the target address of a Jump opcode.
- bool setFunctionStackDeclaration (size_t opcodeAddress, uinteger_t argc, uinteger_t targetPC)

    Set the stack details of a function declaration.
- void pushEnvironment (const std::shared_ptr< AstNode > &ast)

    Create a new compile/execute environment stack entry.
- void popEnvironment ()

    Remove a compile/execute environment stack entry.
- void addIdentifier (const std::string &name, std::optional< size_t > position={})

    Add an identifier to the environment.
- const std::map< std::string, size_t > & getIdentifiers () const

    Get the identifier map of the current environment.
- void addIdentifierAssigned (const std::string &name)

    Indicate that an identifier will be altered within the associated scope.
- const std::set< std::string > & getIdentifiersAssigned () const

    Get the set of identifiers that will be assigned in the current scope.
- void addString (const std::string &name)

    Add a string to the environment.
- const std::map< std::string, size_t > & getStrings () const

    Get the string map of the current environment.
- void pushBreakStack ()

    Increase the `break` environment stack, so that we can handle nested break-supporting structures.
- void addBreak (size_t location)

    Add the Bytecode location of a `break` statement, to be set when the final target is known at a later time.
- void popBreakStack (size_t target)

    For all `continue` bytecode locations collected by Tang::addContinue, set the target pc to `target`.
- void pushContinueStack ()

Increase the `continue` environment stack, so that we can handle nested continue-supporting structures.

- void addContinue (size_t location)

  Add the Bytecode location of a `continue` statement, to be set when the final target is known at a later time.
- void popContinueStack (size_t target)

  For all `continue` bytecode locations collected by Tang::addContinue, set the target pc to `target`.

Public Attributes

- std::string out

  The output of the program, resulting from the program execution.
- std::vector< std::set< std::string > > functionsCollected

  Names of the functions that are declared in a previous or the current scope.
- std::map< std::string, std::pair< uinteger_t, uinteger_t > > functionsDeclared

  Key/value pair of the function declaration information.
- std::map< std::string, std::vector< Tang::uinteger_t > > functionStackDeclarations

  For each function name, a list of Bytecode addresses that need to be replaced by a function definition.

Private Member Functions

- void parse ()

  Parse the code into an AST.
- void compile ()

  Compile the AST into Bytecode.

Private Attributes

- std::vector< std::map< std::string, size_t > > identifierStack

  Stack of mappings of identifiers to their stack locations.
- std::vector< std::set< std::string > > identifiersAssignedStack

  Stack of sets of identifiers that are the target of an assignment statement within the associated scope.
- std::vector< std::map< std::string, size_t > > stringStack

  Stack of mappings of strings to their stack locations.
- std::vector< std::set< size_t > > breakStack

  Stack of a collection of `break` statement locations.
- std::vector< std::set< size_t > > continueStack

  Stack of a collection of `continue` statement locations.
- std::string code

  The code supplied when the Program was instantiated.
- CodeType codeType

  The type of code that was supplied when the Program was instantiated.
- shared_ptr< AstNode > ast

  A pointer to the AST, if parsing was successful.
- Bytecode bytecode

  The Bytecode of the compiled program.
- std::optional< GarbageCollected > result

  The result of the Program execution.

5.43.1 Detailed Description

Represents a compiled script or template that may be executed.

5.43.2 Member Enumeration Documentation

5.43.2.1 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.

5.43.3 Constructor & Destructor Documentation

5.43.3.1 Program()

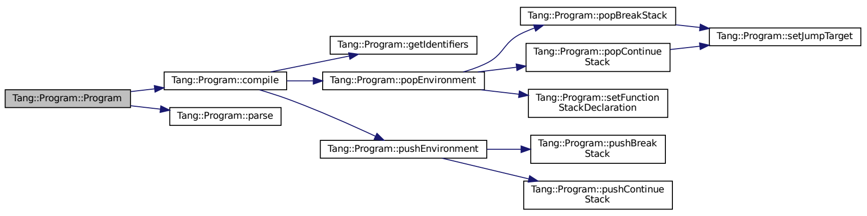
```
Program::Program (
            std::string code,
            Program::CodeType codeType )
```

Create a compiled program using the provided code.

Parameters

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

Here is the call graph for this function:


Tang::Program::Program
Tang::Program::compile
Tang::Program::parse
Tang::Program::getIdentifiers
Tang::Program::popEnvironment
Tang::Program::pushEnvironment
Tang::Program::popBreakStack
Tang::Program::popContinue Stack
Tang::Program::setFunction StackDeclaration
Tang::Program::pushBreak Stack
Tang::Program::pushContinue Stack
Tang::Program::setJumpTarget

5.43.4 Member Function Documentation

5.43.4.1 addBreak()

```
void Program::addBreak (
            size_t location )
```

Add the Bytecode location of a `break` statement, to be set when the final target is known at a later time.

Parameters

location	The offset location of the `break` bytecode.

5.43.4.2 addBytecode()

```
size_t Program::addBytecode (
            Tang::uinteger_t op )
```

Add a Tang::uinteger_t to the Bytecode.

Parameters

op	The value to add to the Bytecode.

Returns

The size of the bytecode structure.

5.43.4.3 addContinue()

```
void Program::addContinue (
            size_t location )
```

Add the Bytecode location of a `continue` statement, to be set when the final target is known at a later time.

Parameters

location	The offset location of the `continue` bytecode.

5.43.4.4 addIdentifier()

```
void Program::addIdentifier (
            const std::string & name,
            std::optional< size_t > position = {} )
```

Add an identifier to the environment.

Parameters

name	The variable to add to the environment.
position	If provided, the desired position to place the identifier.

5.43.4.5 addIdentifierAssigned()

```
void Program::addIdentifierAssigned (
            const std::string & name )
```

Indicate that an identifier will be altered within the associated scope.

Parameters

name	The identifier name.

5.43.4.6 addString()

```
void Program::addString (
            const std::string & name )
```

Add a string to the environment.

Parameters

name	The variable to add to the environment.
position	If provided, the desired position to place the identifier.

5.43.4.7 dumpBytecode()

`string Program::dumpBytecode ( ) const`

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

Returns

A string containing the Opcode representation.

5.43.4.8 execute()

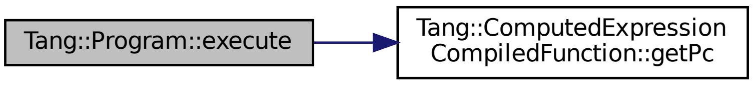
`Program & Program::execute ( )`

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

Returns

The current Program object.

Here is the call graph for this function:



5.43.4.9 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.

5.43.4.10 getBytecode()

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

Get the Bytecode vector.

Returns

The Bytecode vector.

5.43.4.11 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.

5.43.4.12 getIdentifiers()

```
const map< string, size_t > & Program::getIdentifiers ( ) const
```

Get the identifier map of the current environment.

Returns

A map of each identifer name to its stack position within the current environment.

5.43.4.13 getIdentifiersAssigned()

```
const set< string > & Program::getIdentifiersAssigned ( ) const
```

Get the set of identifiers that will be assigned in the current scope.

Returns

A set of identifier names that have been identified as the target of an assignment operator within the current scope.

5.43.4.14 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.

5.43.4.15 getStrings()

```
const map< string, size_t > & Program::getStrings ( ) const
```

Get the string map of the current environment.

Returns

A map of each identifer name to its stack position within the current environment.

5.43.4.16 popBreakStack()

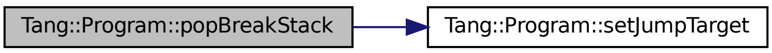
```
void Program::popBreakStack (
             size_t target )
```

For all `continue` bytecode locations collected by Tang::addContinue, set the target pc to `target`.

Parameters

target	The target bytecode offset that the `continue` should jump to.

Here is the call graph for this function:



5.43.4.17 popContinueStack()

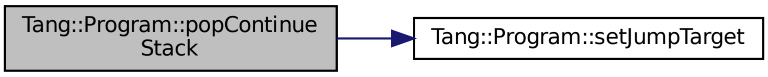
```
void Program::popContinueStack (
             size_t target )
```

For all `continue` bytecode locations collected by Tang::addContinue, set the target pc to `target`.

Parameters

target	The target bytecode offset that the `continue` should jump to.

Here is the call graph for this function:


Tang::Program::popContinue Stack
Tang::Program::setJumpTarget

5.43.4.18 pushEnvironment()

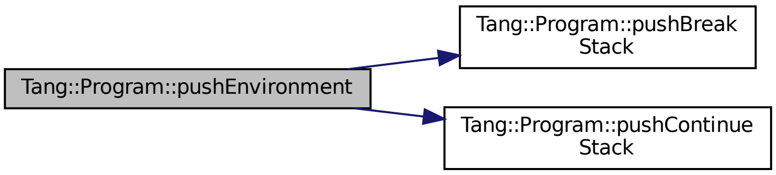
```
void Program::pushEnvironment (
             const std::shared_ptr< AstNode > & ast )
```

Create a new compile/execute environment stack entry.

Parameters

ast	The ast node from which this new environment will be formed.

Here is the call graph for this function:


Tang::Program::pushBreak Stack
Tang::Program::pushEnvironment
Tang::Program::pushContinue Stack

5.43.4.19 setFunctionStackDeclaration()

```
bool Program::setFunctionStackDeclaration (
             size_t opcodeAddress,
             uinteger_t argc,
             uinteger_t targetPC )
```

Set the stack details of a function declaration.

Parameters

opcodeAddress	The location of the FUNCTION opcode.
argc	The argument count to set.
targetPC	The bytecode address of the start of the function.

5.43.4.20 setJumpTarget()

```
bool Program::setJumpTarget (
             size_t opcodeAddress,
             Tang::uinteger_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.

5.43.5 Member Data Documentation

5.43.5.1 functionsDeclared

```
std::map<std::string, std::pair<uinteger_t, uinteger_t> > Tang::Program::functionsDeclared
```

Key/value pair of the function declaration information.

The key is the name of the function. The value is a `pair` of the `argc` value and the `targetPC` value.

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

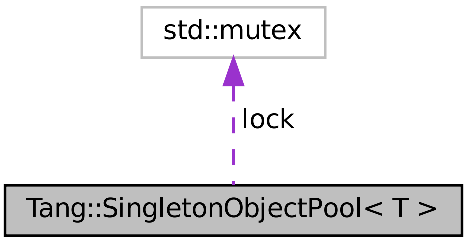
- include/program.hpp
- src/program-dumpBytecode.cpp
- src/program-execute.cpp
- src/program.cpp

5.44 Tang::SingletonObjectPool< T > Class Template Reference

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

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

Collaboration diagram for Tang::SingletonObjectPool< T >:



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.

Private Member Functions

- SingletonObjectPool ()

  The constructor, hidden from being directly called.
- SingletonObjectPool (const SingletonObjectPool &other)

  The copy constructor, hidden from being called.

Private Attributes

- T ∗∗ allocations

  C-array of allocated blocks, each block contains `GROW` objects.
- int currentAllocation

  Index into `allocations`, representing the current block supplying non-recycled memory addresses.
- size_t currentIndex

  Current location (within the most recently allocated block) of an available T∗.
- int currentRecycledAllocation

  Index into `allocations`, representing the current block tracking the recycled memory addresses.
- int currentRecycledIndex

  Current location (within the `currentRecycledAllocation` block) of the last available T∗.

Static Private Attributes

- static std::mutex lock

  A mutex for thread-safety.

5.44.1 Detailed Description

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

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

5.44.2 Member Function Documentation

5.44.2.1 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.

5.44.2.2 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.

5.44.2.3 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.

5.44.3 Member Data Documentation

5.44.3.1 currentIndex

```
template<class T >
size_t Tang::SingletonObjectPool< T >::currentIndex  [private]
```

Current location (within the most recently allocated block) of an available T$*$.

If currentIndex == GROW, then a new block needs to be allocated.

5.44.3.2 currentRecycledIndex

```
template<class T >
int Tang::SingletonObjectPool< T >::currentRecycledIndex  [private]
```

Current location (within the `currentRecycledAllocation` block) of the last available T$*$.

If currentRecycledIndex == GROW, then we must move to the next currentRecycledAllocation.

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

- include/singletonObjectPool.hpp

5.45 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.

5.45.1 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).

5.45.2 Constructor & Destructor Documentation

5.45.2.1 TangBase()

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

The constructor.

Isn't it glorious.

5.45.3 Member Function Documentation

5.45.3.1 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:

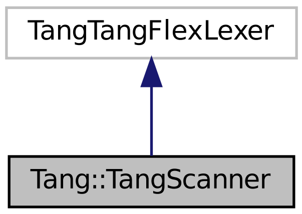
- include/tangBase.hpp
- src/tangBase.cpp

5.46 Tang::TangScanner Class Reference

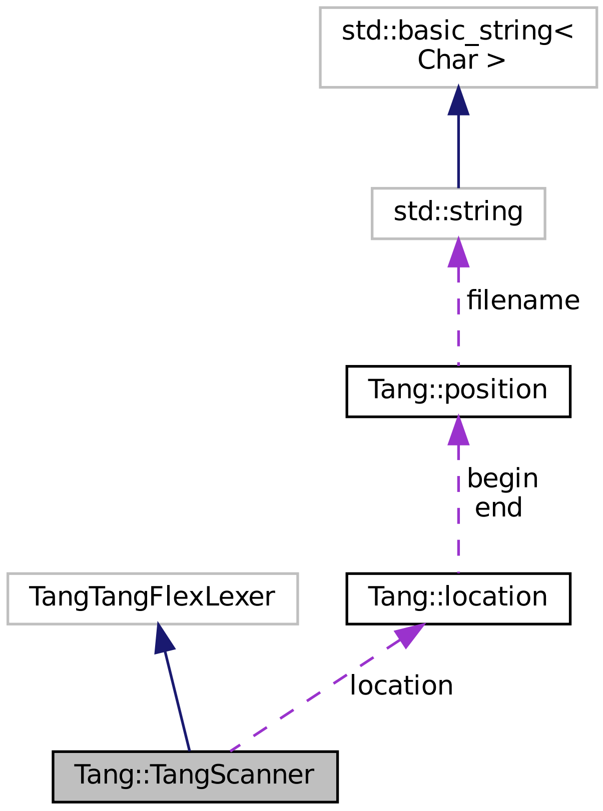
The Flex lexer class for the main Tang language.

```
#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.

Private Attributes

- Tang::location location

    The location information of the token that is identified.

5.46.1 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.

5.46.2 Constructor & Destructor Documentation

5.46.2.1 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)

5.46.3 Member Function Documentation

5.46.3.1 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:

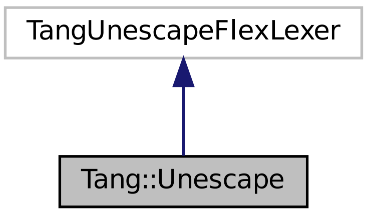
- include/tangScanner.hpp

5.47 Tang::Unescape Class Reference

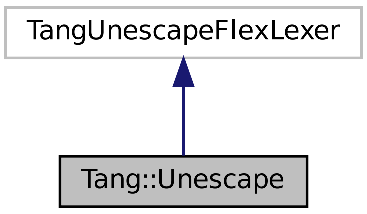
The Flex lexer class for the main Tang language.

```
#include <unescape.hpp>
```

Inheritance diagram for Tang::Unescape:



Collaboration diagram for Tang::Unescape:



Public Member Functions

- Unescape (std::istream &arg_yyin, std::ostream &arg_yyout)

  The constructor for the Scanner.
- virtual std::string get_next_token ()

  Extract the next token from the input string.

5.47.1 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.

5.47.2 Constructor & Destructor Documentation

5.47.2.1 Unescape()

```
Tang::Unescape::Unescape (
            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)

5.47.3 Member Function Documentation

5.47.3.1 get_next_token()

```
virtual std::string Tang::Unescape::get_next_token ( )  [virtual]
```

Extract the next token from the input string.

Returns

The next unescaped character.

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

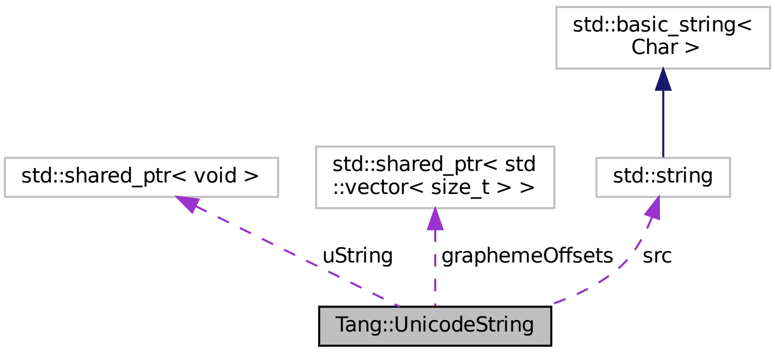
- include/unescape.hpp

5.48 Tang::UnicodeString Class Reference

Represents a UTF-8 encoded string that is Unicode-aware.

```
#include <unicodeString.hpp>
```

Collaboration diagram for Tang::UnicodeString:



Public Member Functions

- UnicodeString (const std::string &src)

  Construct a Tang::UnicodeString object, which acts as the interface to the ICU library.
- std::string substr (size_t position, size_t length) const

  Return a Unicode grapheme-aware substring.
- bool operator== (const UnicodeString &rhs) const

  Compare two UnicodeStrings.
- bool operator< (const UnicodeString &rhs) const

  Compare two UnicodeStrings.
- UnicodeString operator+ (const UnicodeString &rhs) const

  Create a new UnicodeString that is the concatenation of two UnicodeStrings.
- operator std::string () const

  Cast the current UnicodeString object to a std::string, UTF-8 encoded.
- size_t length () const

  Return the length of the UnicodeString in graphemes.
- size_t bytesLength () const

  Return the length of the UnicodeString in bytes.

Private Member Functions

- void generateCachedValues () const

  Calculate cachable values for the object.

Private Attributes

- std::string src

    The UTF-8 encoded string.
- std::shared_ptr< std::vector< size_t > > graphemeOffsets

    Cache of the grapheme offsets, if they happen to be calculated.
- std::shared_ptr< void > uString

    Cache of the ICU Unicode string.

5.48.1 Detailed Description

Represents a UTF-8 encoded string that is Unicode-aware.

This class serves as the interface between the Tang language and the ICU library.

5.48.2 Constructor & Destructor Documentation

5.48.2.1 UnicodeString()

```
UnicodeString::UnicodeString (
            const std::string & src )
```

Construct a Tang::UnicodeString object, which acts as the interface to the ICU library.

Parameters

src	A UTF-8 encoded string.

5.48.3 Member Function Documentation

5.48.3.1 bytesLength()

```
size_t UnicodeString::bytesLength ( ) const
```

Return the length of the UnicodeString in bytes.

Note: this is *not* the number of codepoints or graphemes, but is the acutal number of bytes in memory.

Returns

Returns the length of the UnicodeString in bytes.

5.48.3.2 length()

```
size_t UnicodeString::length ( ) const
```

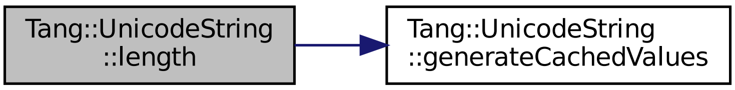
Return the length of the UnicodeString in graphemes.

Note: this is *not* the number of bytes, chars, or codepoints, but is the length in graphemes, as defined by ICU.

Returns

Returns the length of the UnicodeString in graphemes.

Here is the call graph for this function:


Tang::UnicodeString ::length
Tang::UnicodeString ::generateCachedValues

5.48.3.3 operator std::string()

```
UnicodeString::operator std::string ( ) const
```

Cast the current UnicodeString object to a std::string, UTF-8 encoded.

Returns

Returns the std::string version of the UnicodeString.

5.48.3.4 operator+()

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

Create a new UnicodeString that is the concatenation of two UnicodeStrings.

Parameters

rhs	The string to append to the current object string.

Returns

Returns the result of the concatenation.

5.48.3.5 operator<()

```
bool UnicodeString::operator< (
            const UnicodeString & rhs ) const
```

Compare two UnicodeStrings.

Parameters

rhs	The string to compare against.

Returns

Returns true if the rhs string is greater than or equal to the object string.

5.48.3.6 operator==()

```
bool UnicodeString::operator== (
            const UnicodeString & rhs ) const
```

Compare two UnicodeStrings.

Parameters

rhs	The string to compare against.

Returns

Returns true if the two strings are equal.

5.48.3.7 substr()

```
std::string UnicodeString::substr (
            size_t position,
            size_t length ) const
```

Return a Unicode grapheme-aware substring.

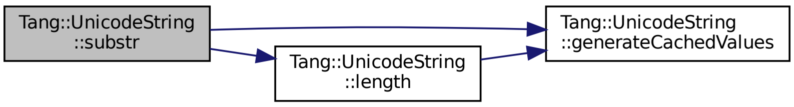
Parameters

position	The 0-based position of the first grapheme.
length	The maximum number of graphemes to return.

Returns

The requested substring.

Here is the call graph for this function:


Tang::UnicodeString ::substr
Tang::UnicodeString ::length
Tang::UnicodeString ::generateCachedValues

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

- include/unicodeString.hpp
- src/unicodeString.cpp

Chapter 6

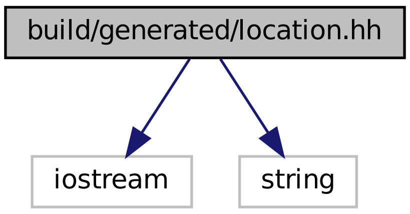
File Documentation

6.1 build/generated/location.hh File Reference

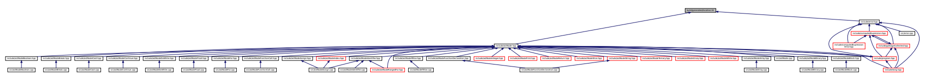
Define the Tang ::location class.

```
#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

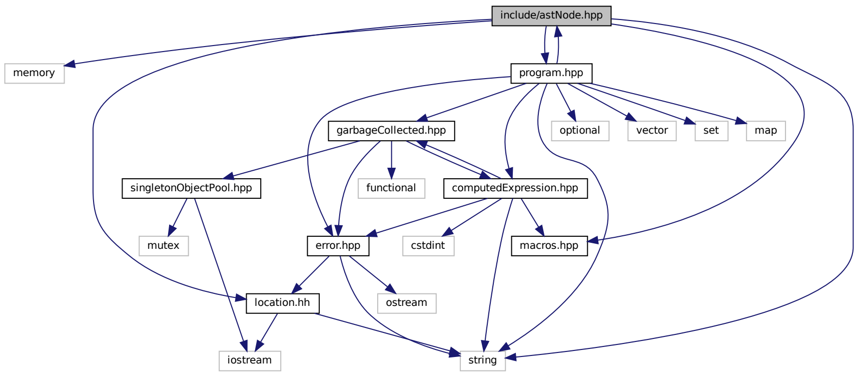
ostr	the destination output stream
pos	a reference to the position to redirect

6.2 include/astNode.hpp File Reference

Declare the Tang::AstNode base class.

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

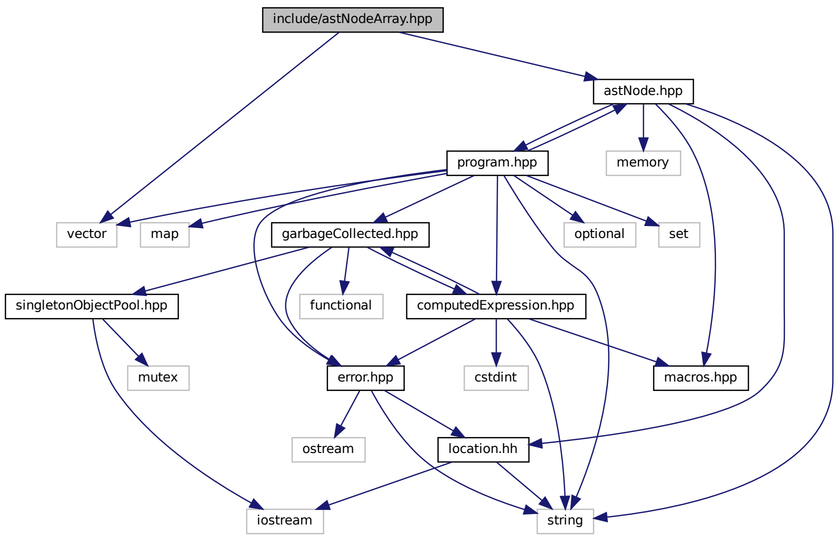
Declare the Tang::AstNode base class.

6.3 include/astNodeArray.hpp File Reference

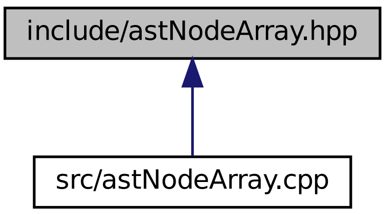
Declare the Tang::AstNodeArray class.

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

Include dependency graph for astNodeArray.hpp:



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



Classes

- class Tang::AstNodeArray

    An AstNode that represents an array literal.

6.3.1 Detailed Description

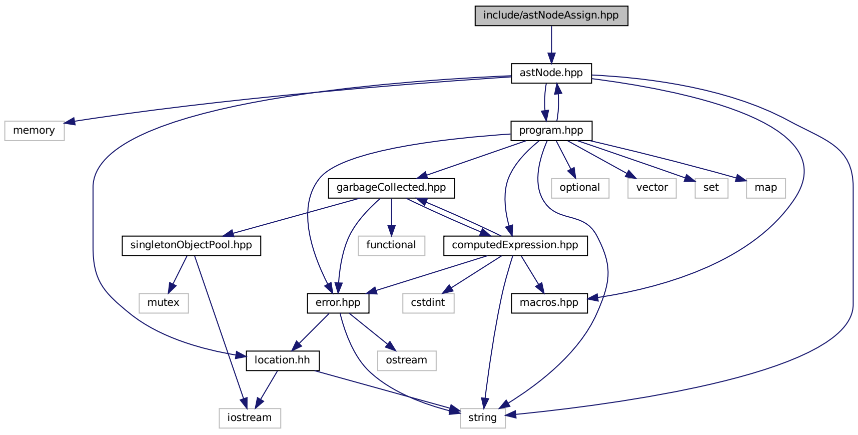
Declare the Tang::AstNodeArray class.

6.4 include/astNodeAssign.hpp File Reference

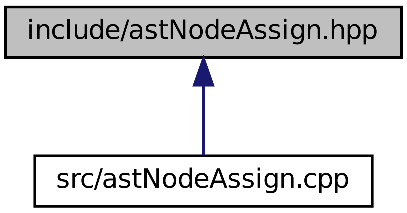
Declare the Tang::AstNodeAssign class.

```
#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.

6.4.1 Detailed Description

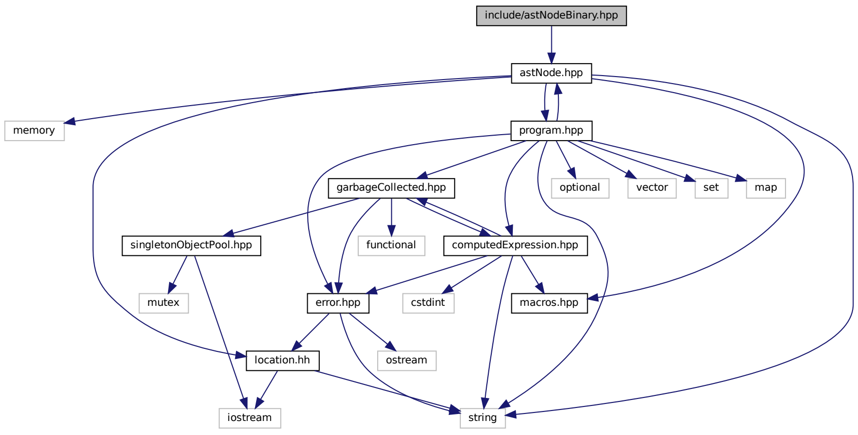
Declare the Tang::AstNodeAssign class.

6.5 include/astNodeBinary.hpp File Reference

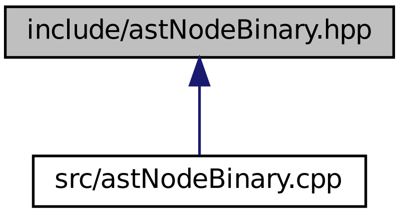
Declare the Tang::AstNodeBinary class.

```
#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.

6.5.1 Detailed Description

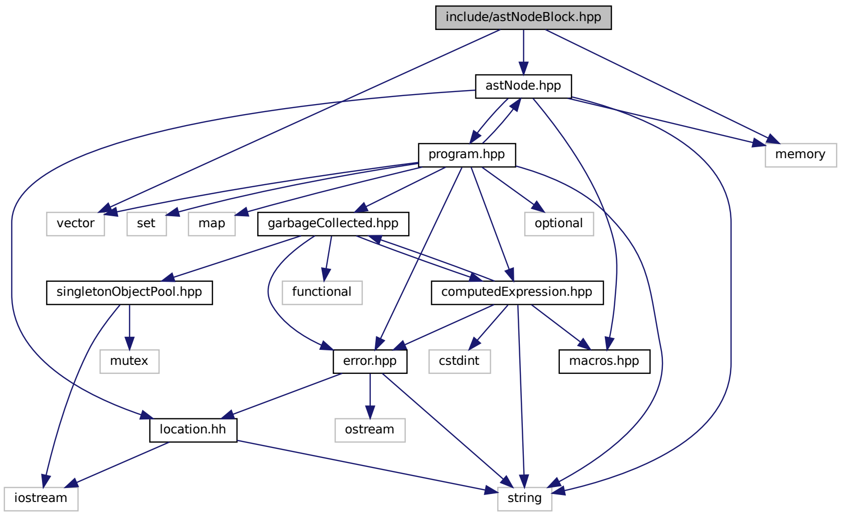
Declare the Tang::AstNodeBinary class.

6.6 include/astNodeBlock.hpp File Reference

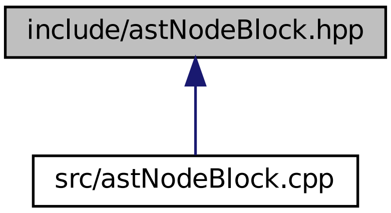
Declare the Tang::AstNodeBlock class.

```
#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.

6.6.1 Detailed Description

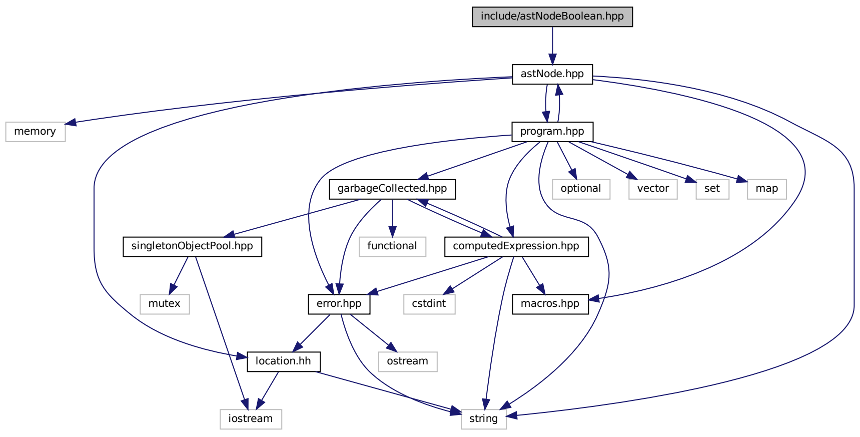
Declare the Tang::AstNodeBlock class.

6.7 include/astNodeBoolean.hpp File Reference

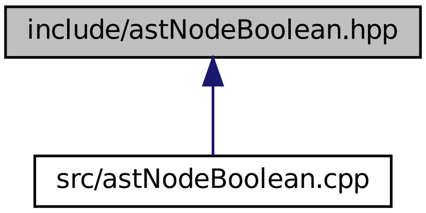
Declare the Tang::AstNodeBoolean class.

```
#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.

6.7.1 Detailed Description

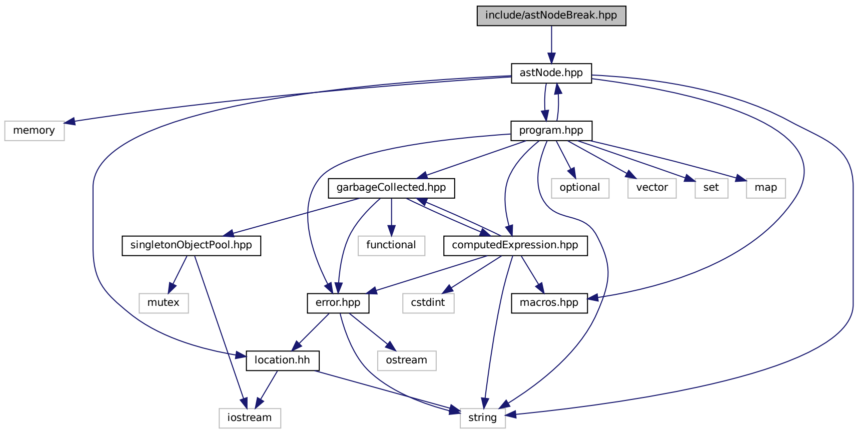
Declare the Tang::AstNodeBoolean class.

6.8 include/astNodeBreak.hpp File Reference

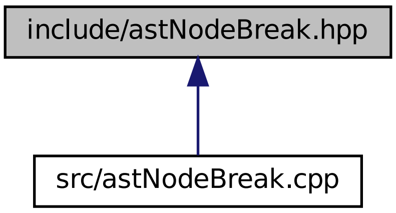
Declare the Tang::AstNodeBreak class.

`#include "astNode.hpp"`

Include dependency graph for astNodeBreak.hpp:



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



Classes

- class Tang::AstNodeBreak

  An AstNode that represents a `break` statement.

6.8.1 Detailed Description

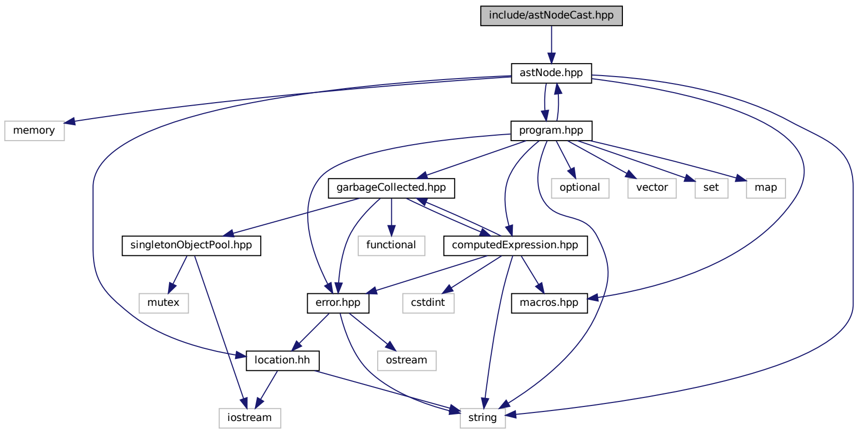
Declare the Tang::AstNodeBreak class.

6.9 include/astNodeCast.hpp File Reference

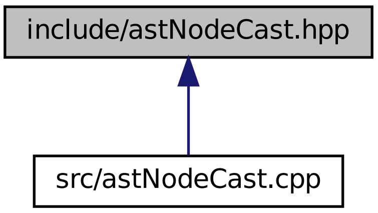
Declare the Tang::AstNodeCast class.

```
#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.

6.9.1 Detailed Description

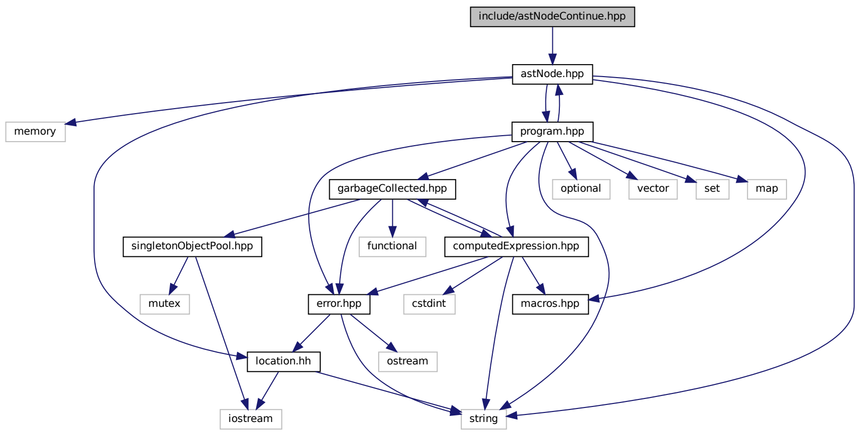
Declare the Tang::AstNodeCast class.

6.10 include/astNodeContinue.hpp File Reference

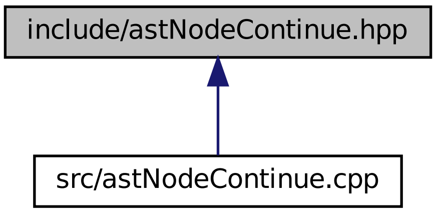
Declare the Tang::AstNodeContinue class.

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

Include dependency graph for astNodeContinue.hpp:



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



Classes

- class Tang::AstNodeContinue

  An AstNode that represents a `continue` statement.

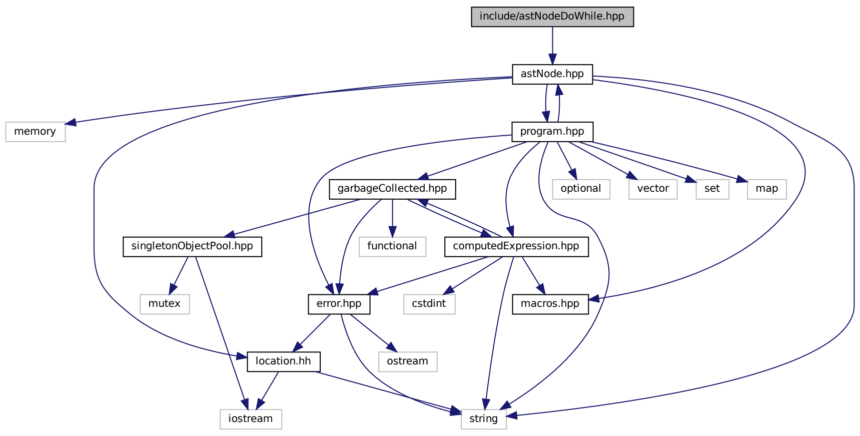
6.10.1 Detailed Description

Declare the Tang::AstNodeContinue class.

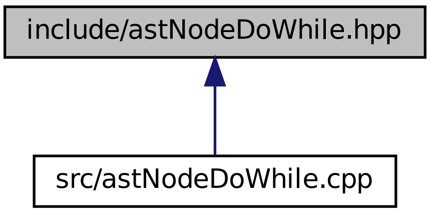
6.11 include/astNodeDoWhile.hpp File Reference

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.

6.11.1 Detailed Description

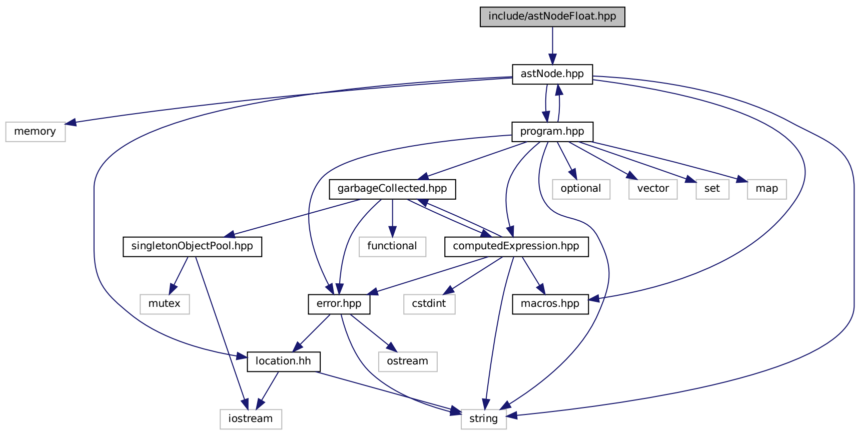
Declare the Tang::AstNodeDoWhile class.

6.12 include/astNodeFloat.hpp File Reference

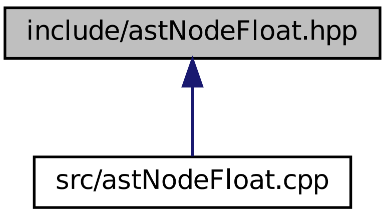
Declare the Tang::AstNodeFloat class.

```
#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.

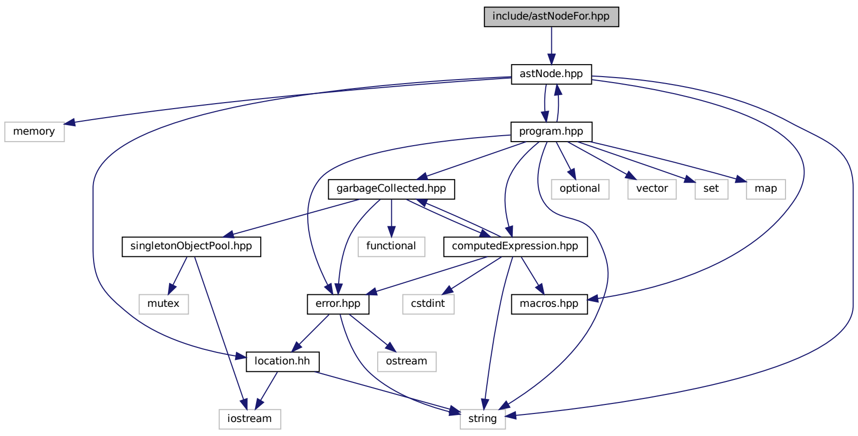
6.12.1 Detailed Description

Declare the Tang::AstNodeFloat class.

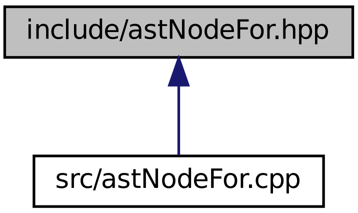
6.13 include/astNodeFor.hpp File Reference

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.

6.13.1 Detailed Description

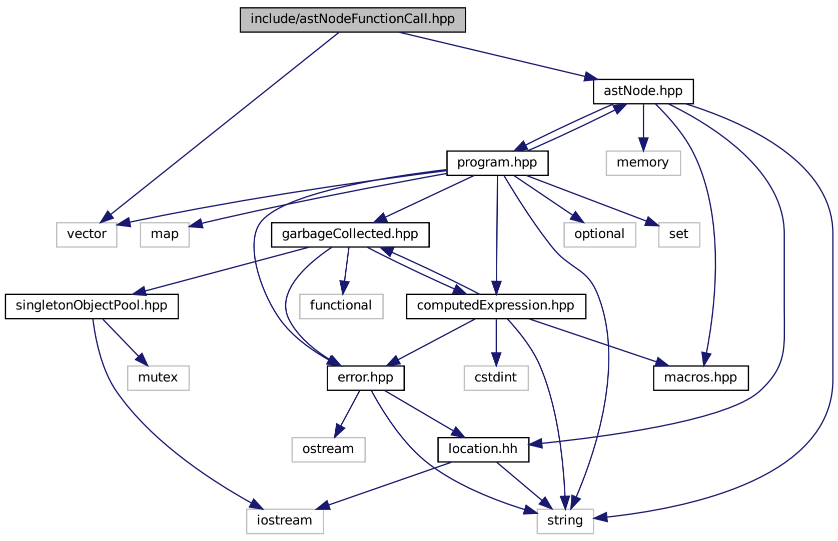
Declare the Tang::AstNodeFor class.

6.14 include/astNodeFunctionCall.hpp File Reference

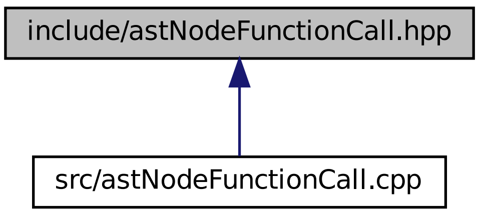
Declare the Tang::AstNodeFunctionCall class.

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

Include dependency graph for astNodeFunctionCall.hpp:



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



Classes

- class Tang::AstNodeFunctionCall

  An AstNode that represents a function call.

6.14.1 Detailed Description

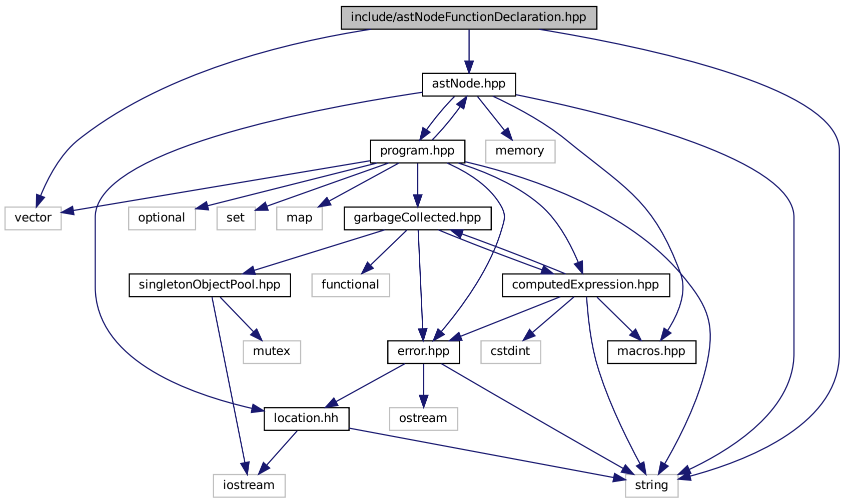
Declare the Tang::AstNodeFunctionCall class.

6.15 include/astNodeFunctionDeclaration.hpp File Reference

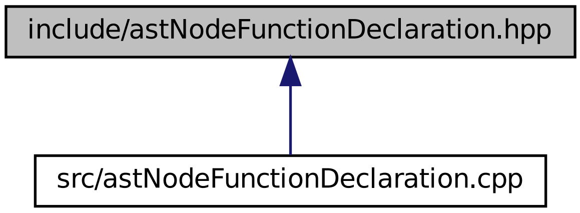
Declare the Tang::AstNodeFunctionDeclaration class.

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

Include dependency graph for astNodeFunctionDeclaration.hpp:



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


include/astNodeFunctionDeclaration.hpp
src/astNodeFunctionDeclaration.cpp

Classes

- class Tang::AstNodeFunctionDeclaration

    An AstNode that represents a function declaration.

6.15.1 Detailed Description

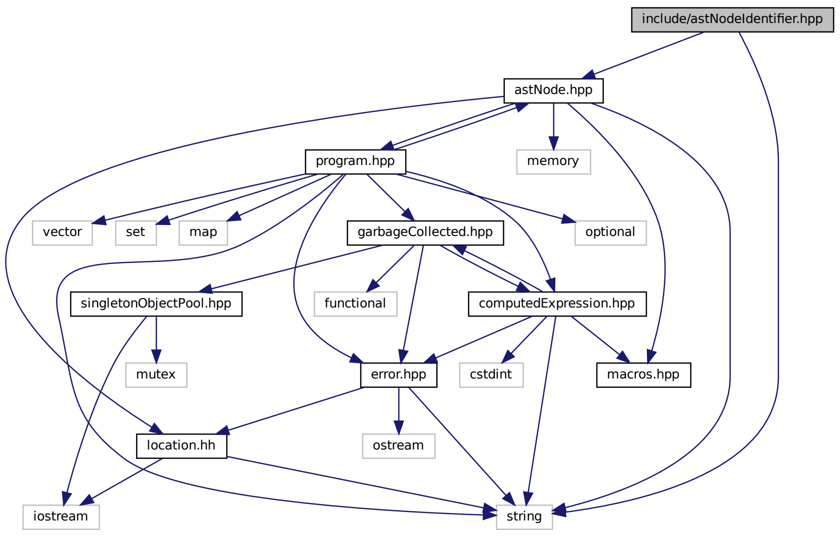
Declare the Tang::AstNodeFunctionDeclaration class.

6.16 include/astNodeIdentifier.hpp File Reference

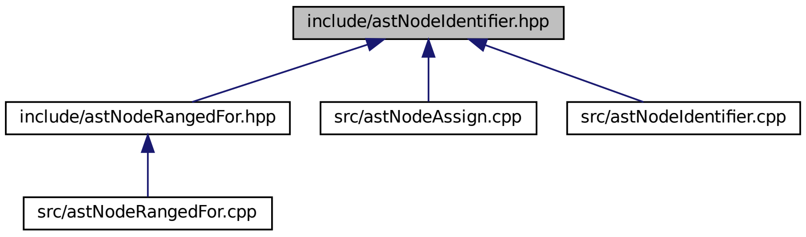
Declare the Tang::AstNodeIdentifier class.

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

Include dependency graph for astNodeIdentifier.hpp:



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


include/astNodeIdentifier.hpp
include/astNodeRangedFor.hpp
src/astNodeAssign.cpp
src/astNodeIdentifier.cpp
src/astNodeRangedFor.cpp

Classes

- class Tang::AstNodeIdentifier

  An AstNode that represents an identifier.

6.16.1 Detailed Description

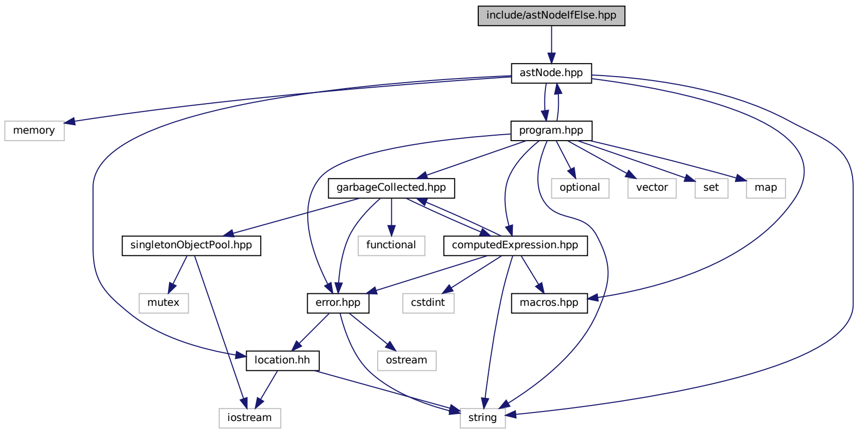
Declare the Tang::AstNodeIdentifier class.

6.17 include/astNodeIfElse.hpp File Reference

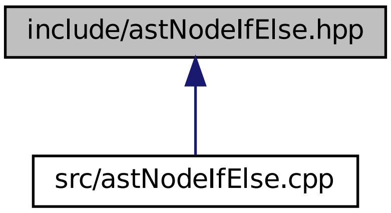
Declare the Tang::AstNodeIfElse class.

```
#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.

6.17.1 Detailed Description

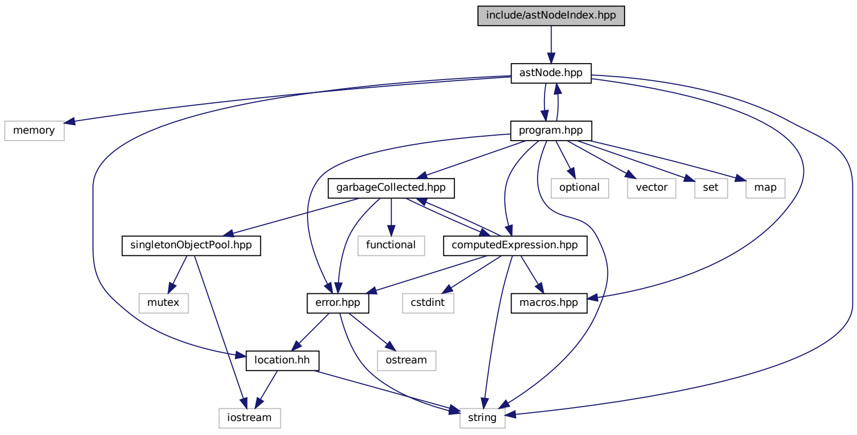
Declare the Tang::AstNodeIfElse class.

6.18 include/astNodeIndex.hpp File Reference

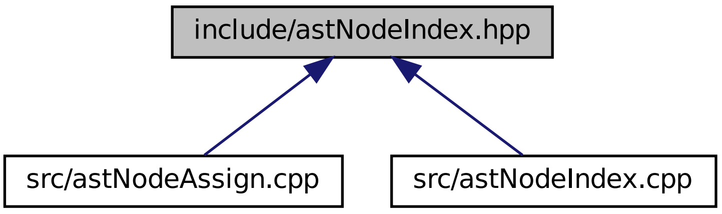
Declare the Tang::AstNodeIndex class.

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

Include dependency graph for astNodeIndex.hpp:



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



Classes

- class Tang::AstNodeIndex

    An AstNode that represents an index into a collection.

6.18.1 Detailed Description

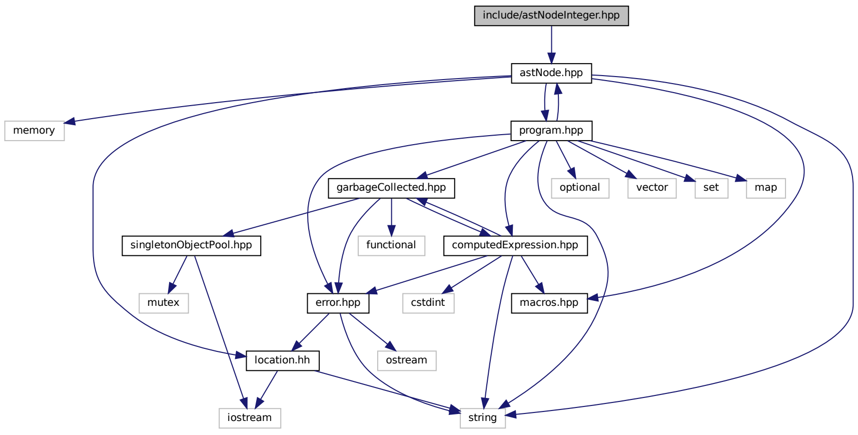
Declare the Tang::AstNodeIndex class.

6.19 include/astNodeInteger.hpp File Reference

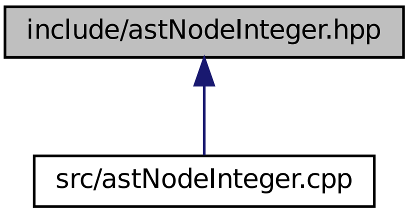
Declare the Tang::AstNodeInteger class.

`#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.

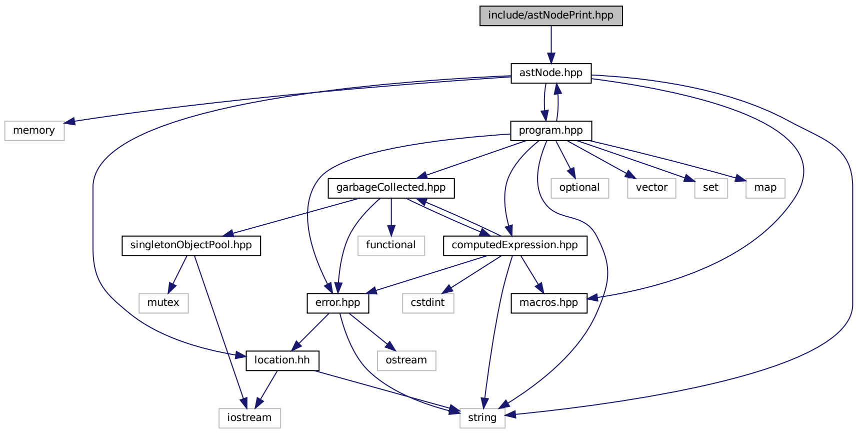
6.19.1 Detailed Description

Declare the Tang::AstNodeInteger class.

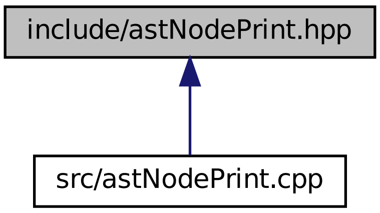
6.20 include/astNodePrint.hpp File Reference

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.

6.20.1 Detailed Description

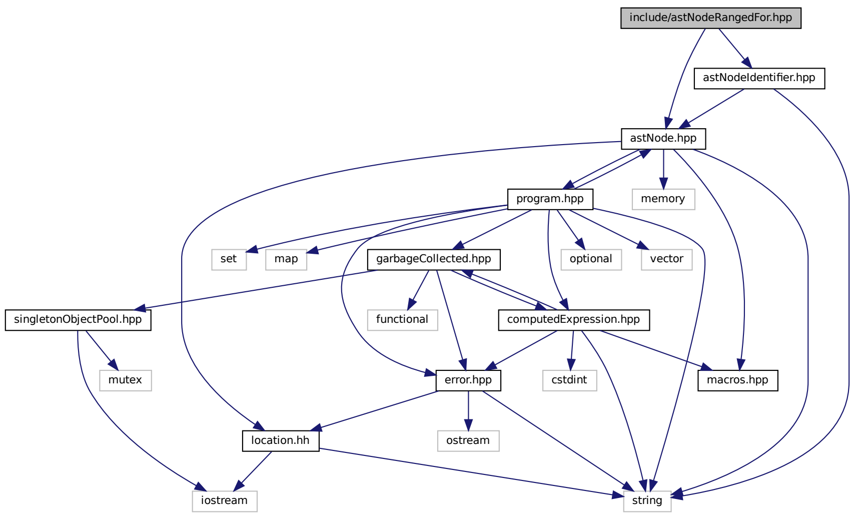
Declare the Tang::AstNodePrint class.

6.21 include/astNodeRangedFor.hpp File Reference

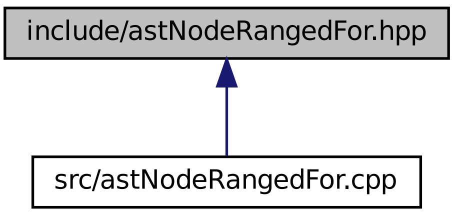
Declare the Tang::AstNodeRangedFor class.

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

Include dependency graph for astNodeRangedFor.hpp:



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



Classes

- class Tang::AstNodeRangedFor

  An AstNode that represents a ranged for() statement.

6.21.1 Detailed Description

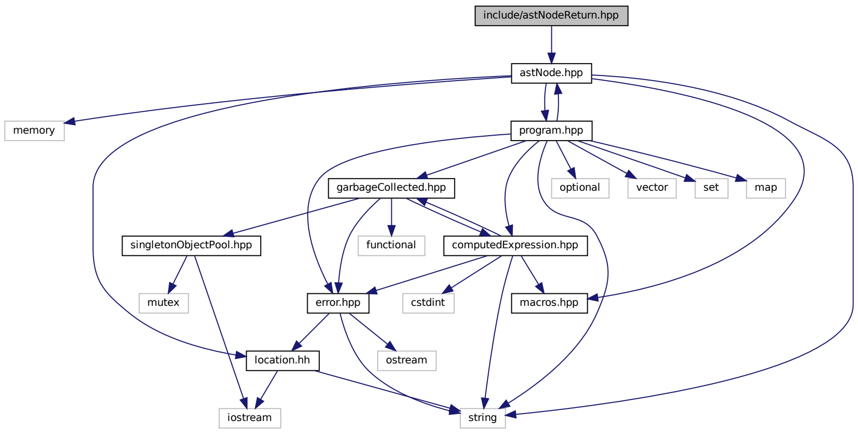
Declare the Tang::AstNodeRangedFor class.

6.22 include/astNodeReturn.hpp File Reference

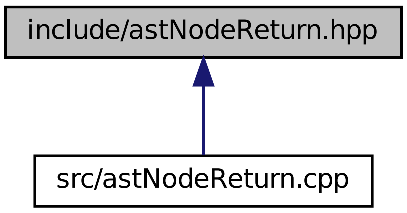
Declare the Tang::AstNodeReturn class.

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

Include dependency graph for astNodeReturn.hpp:



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



Classes

- class Tang::AstNodeReturn

  An AstNode that represents a `return` statement.

6.22.1 Detailed Description

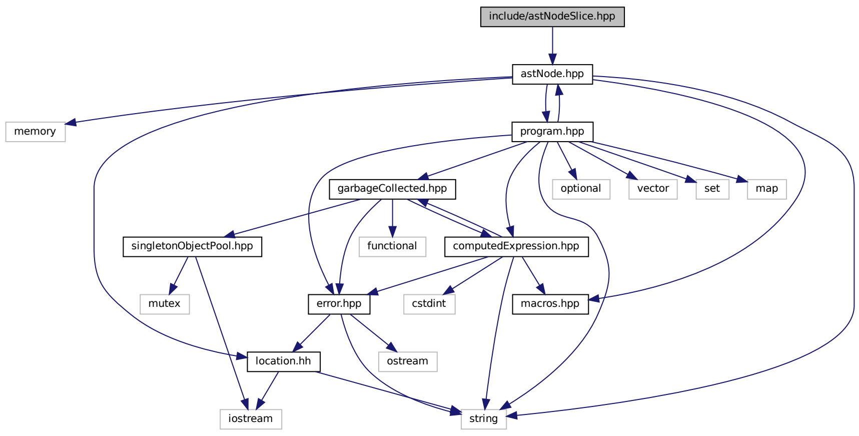
Declare the Tang::AstNodeReturn class.

6.23 include/astNodeSlice.hpp File Reference

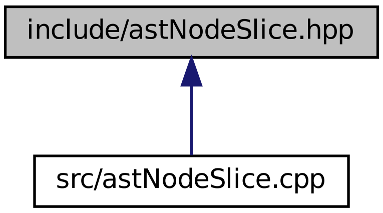
Declare the Tang::AstNodeSlice class.

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

Include dependency graph for astNodeSlice.hpp:



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



Classes

- class Tang::AstNodeSlice

  An AstNode that represents a ternary expression.

6.23.1 Detailed Description

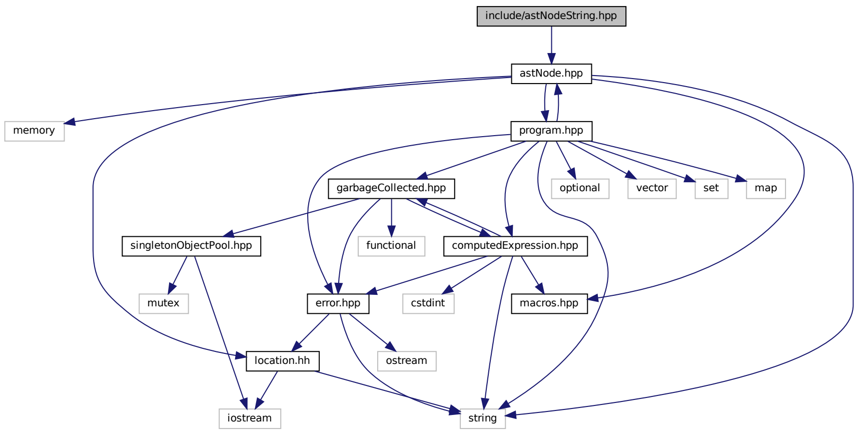
Declare the Tang::AstNodeSlice class.

6.24 include/astNodeString.hpp File Reference

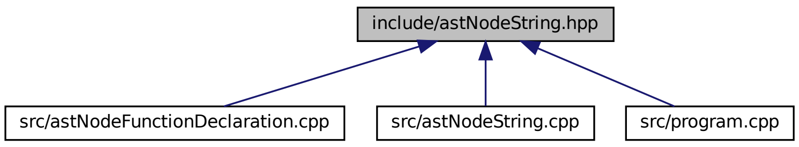
Declare the Tang::AstNodeString class.

`#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.

6.24.1 Detailed Description

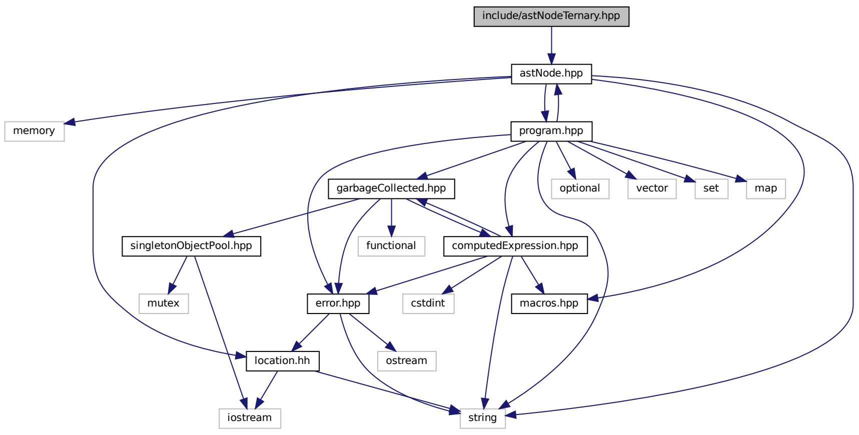
Declare the Tang::AstNodeString class.

6.25 include/astNodeTernary.hpp File Reference

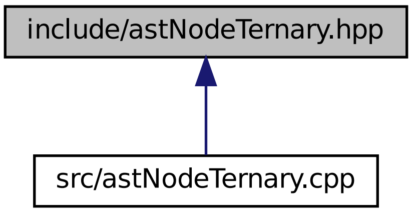
Declare the Tang::AstNodeTernary class.

```
#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.

6.25.1 Detailed Description

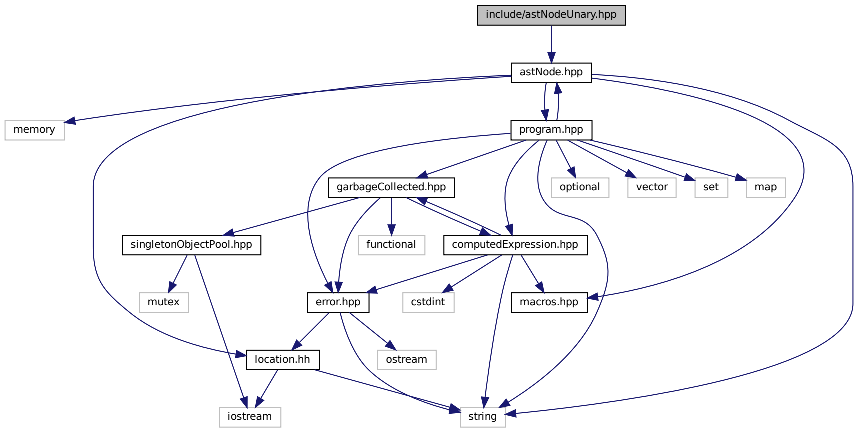
Declare the Tang::AstNodeTernary class.

6.26 include/astNodeUnary.hpp File Reference

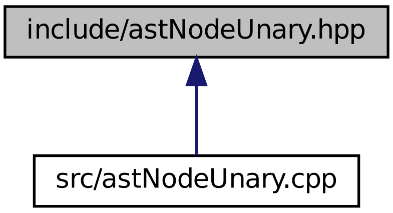
Declare the Tang::AstNodeUnary class.

```
#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.

6.26.1 Detailed Description

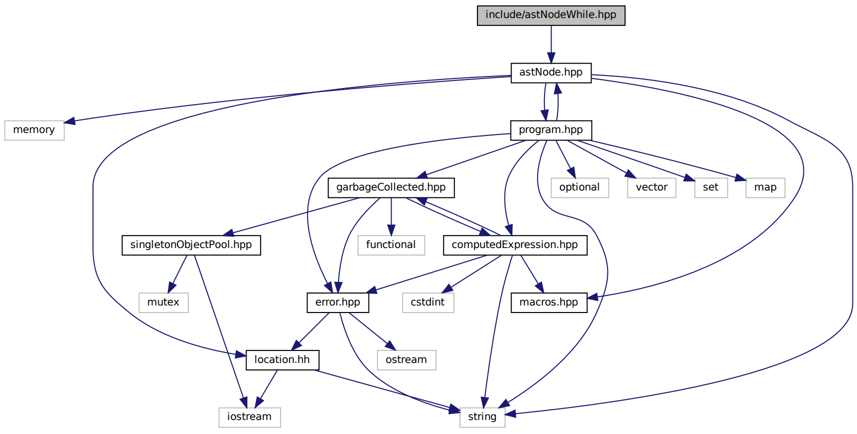
Declare the Tang::AstNodeUnary class.

6.27 include/astNodeWhile.hpp File Reference

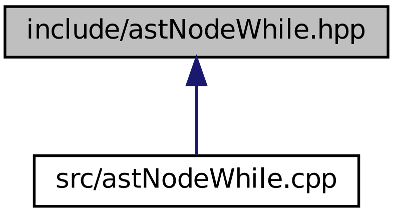
Declare the Tang::AstNodeWhile class.

`#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.

6.27.1 Detailed Description

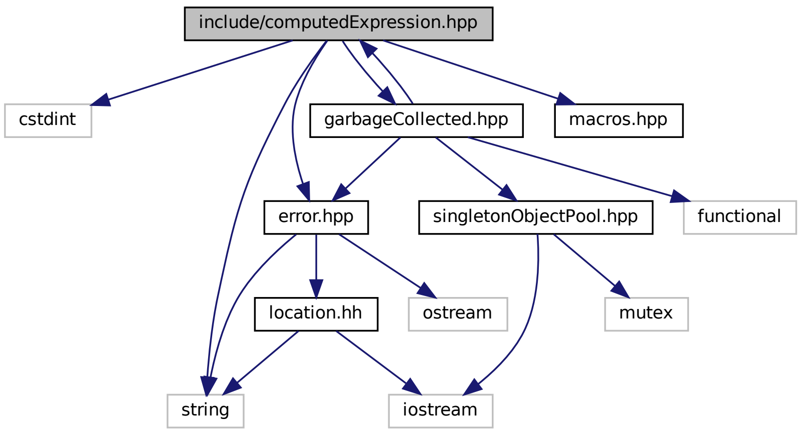
Declare the Tang::AstNodeWhile class.

6.28 include/computedExpression.hpp File Reference

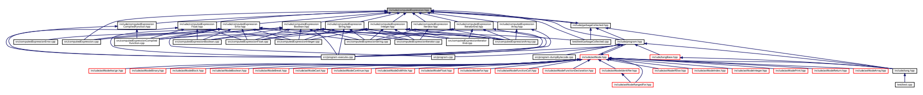
Declare the Tang::ComputedExpression base class.

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

6.28.1 Detailed Description

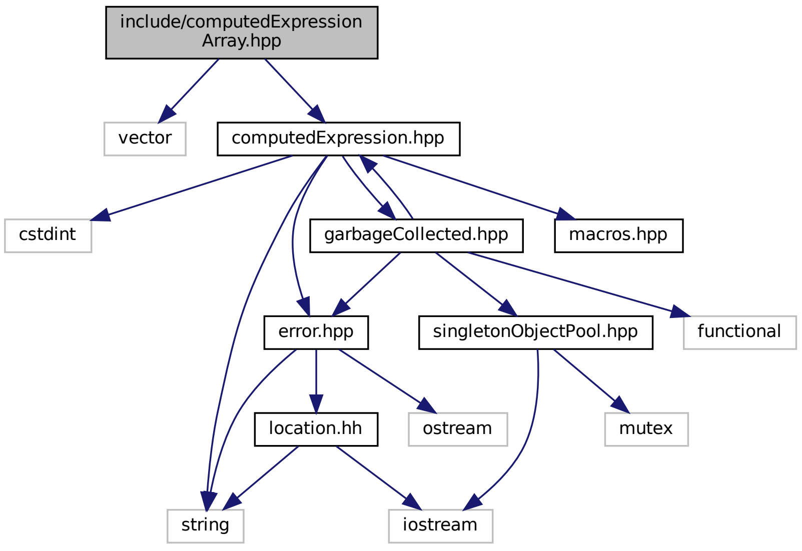
Declare the Tang::ComputedExpression base class.

6.29 include/computedExpressionArray.hpp File Reference

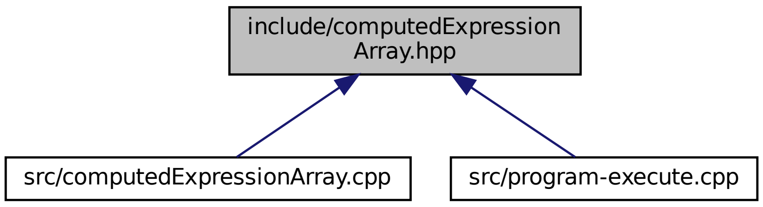
Declare the Tang::ComputedExpressionArray class.

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

Include dependency graph for computedExpressionArray.hpp:



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



Classes

- class Tang::ComputedExpressionArray

  Represents an Array that is the result of a computation.

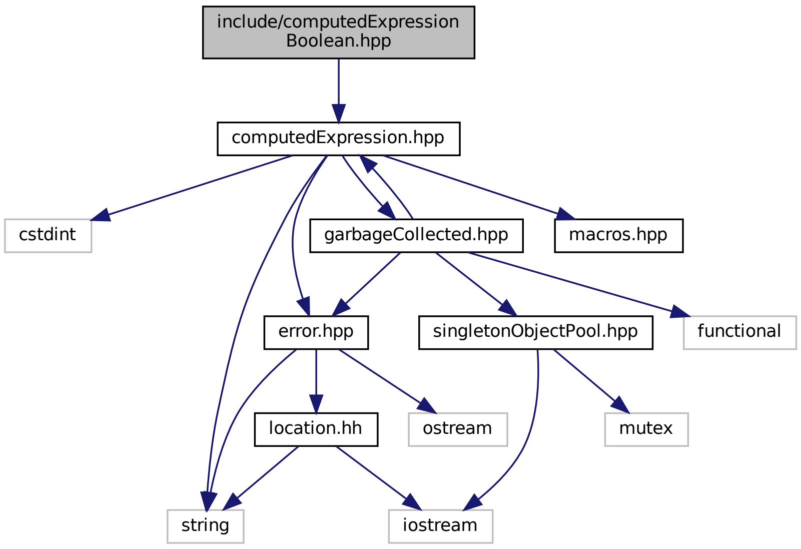
6.29.1 Detailed Description

Declare the Tang::ComputedExpressionArray class.

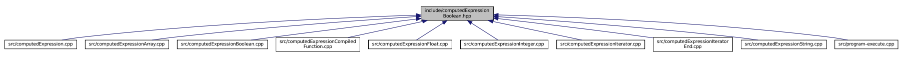
6.30 include/computedExpressionBoolean.hpp File Reference

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/computedExpressionArray.cpp
src/computedExpressionBoolean.cpp
src/computedExpressionCompiled Function.cpp
src/computedExpressionFloat.cpp
src/computedExpressionInteger.cpp
src/computedExpressionIterator.cpp
src/computedExpressionIterator End.cpp
src/computedExpressionString.cpp
src/program-execute.cpp

Classes

- class Tang::ComputedExpressionBoolean

    Represents an Boolean that is the result of a computation.

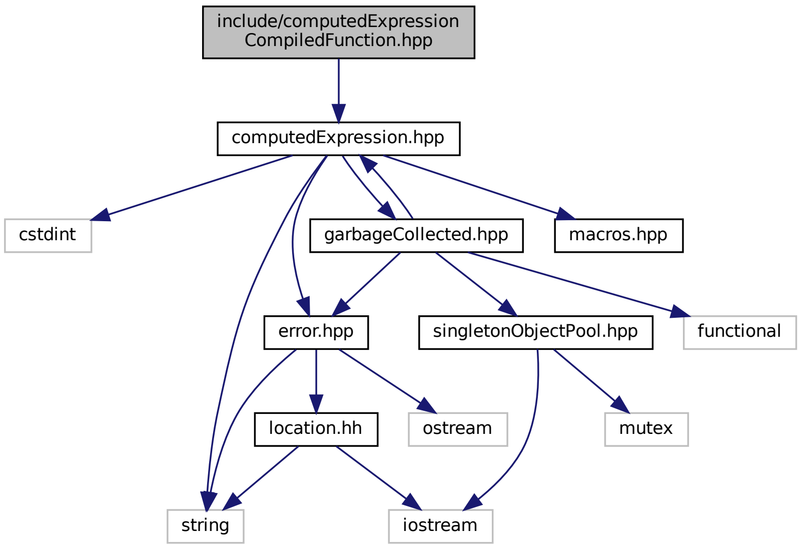
6.30.1 Detailed Description

Declare the Tang::ComputedExpressionBoolean class.

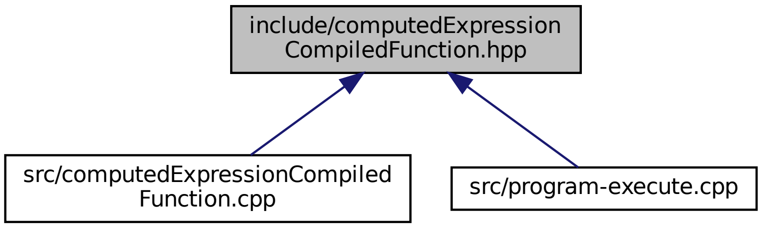
6.31 include/computedExpressionCompiledFunction.hpp File Reference

Declare the Tang::ComputedExpressionCompiledFunction class.

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



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


include/computedExpression CompiledFunction.hpp
src/computedExpressionCompiled Function.cpp
src/program-execute.cpp

Classes

- class Tang::ComputedExpressionCompiledFunction

  Represents a Compiled Function declared in the script.

6.31.1 Detailed Description

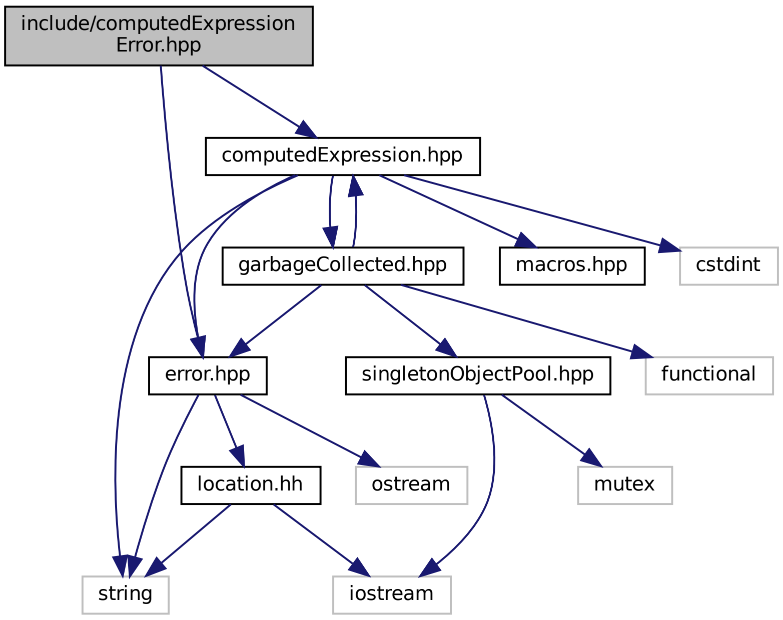
Declare the Tang::ComputedExpressionCompiledFunction class.

6.32 include/computedExpressionError.hpp File Reference

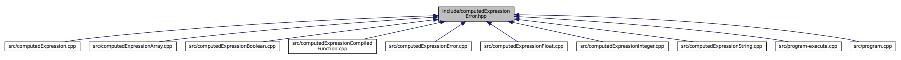
Declare the Tang::ComputedExpressionError class.

```
#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/computedExpressionArray.cpp
src/computedExpressionBoolean.cpp
src/computedExpressionCompiled Function.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.

6.32.1 Detailed Description

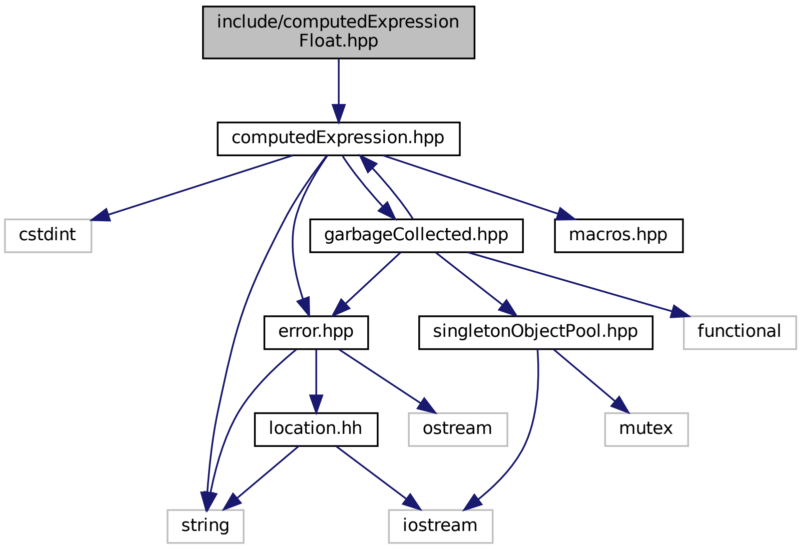
Declare the Tang::ComputedExpressionError class.

6.33 include/computedExpressionFloat.hpp File Reference

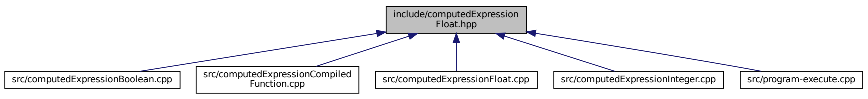
Declare the Tang::ComputedExpressionFloat class.

```
#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/computedExpressionCompiled
Function.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.

6.33.1 Detailed Description

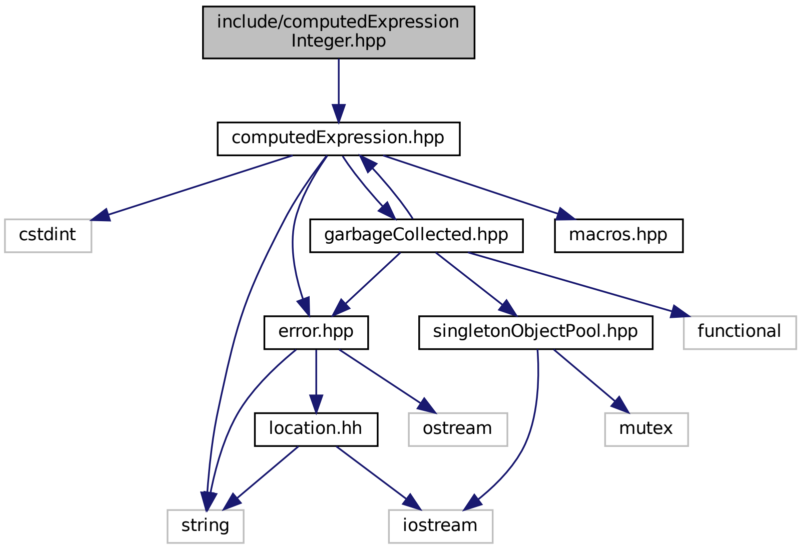
Declare the Tang::ComputedExpressionFloat class.

6.34 include/computedExpressionInteger.hpp File Reference

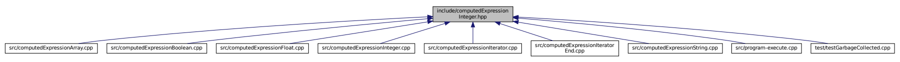
Declare the Tang::ComputedExpressionInteger class.

```
#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/computedExpressionArray.cpp
src/computedExpressionBoolean.cpp
src/computedExpressionFloat.cpp
src/computedExpressionInteger.cpp
src/computedExpressionIterator.cpp
src/computedExpressionIterator
End.cpp
src/computedExpressionString.cpp
src/program-execute.cpp
test/testGarbageCollected.cpp

Classes

- class Tang::ComputedExpressionInteger

  Represents an Integer that is the result of a computation.

6.34.1 Detailed Description

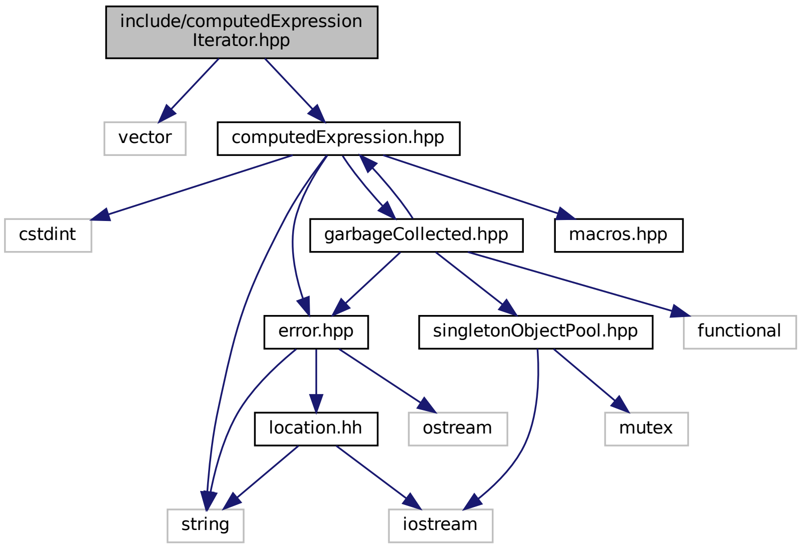
Declare the Tang::ComputedExpressionInteger class.

6.35 include/computedExpressionIterator.hpp File Reference

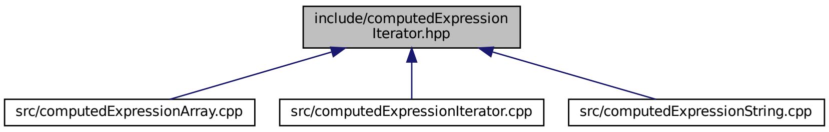
Declare the Tang::ComputedExpressionIterator class.

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

Include dependency graph for computedExpressionIterator.hpp:



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


include/computedExpression Iterator.hpp
src/computedExpressionArray.cpp
src/computedExpressionIterator.cpp
src/computedExpressionString.cpp

Classes

- class Tang::ComputedExpressionIterator

  Represents an Iterator that is the result of a computation.

6.35.1 Detailed Description

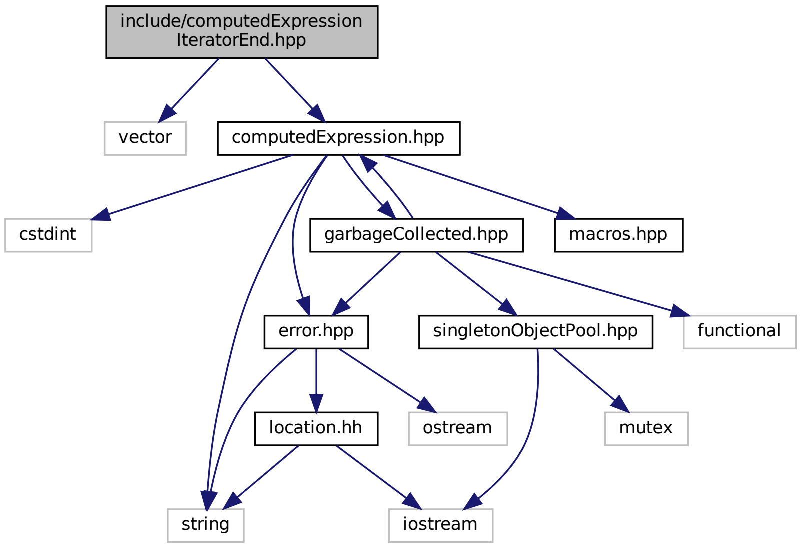
Declare the Tang::ComputedExpressionIterator class.

6.36 include/computedExpressionIteratorEnd.hpp File Reference

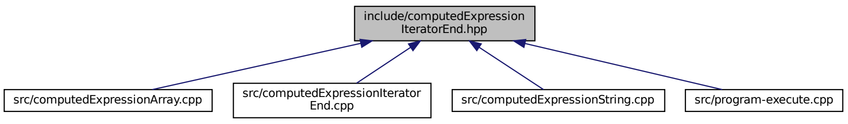
Declare the Tang::ComputedExpressionIteratorEnd class.

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

Include dependency graph for computedExpressionIteratorEnd.hpp:



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


include/computedExpression IteratorEnd.hpp
src/computedExpressionArray.cpp
src/computedExpressionIterator End.cpp
src/computedExpressionString.cpp
src/program-execute.cpp

Classes

- class Tang::ComputedExpressionIteratorEnd

    Represents that a collection has no more values through which to iterate.

6.36.1 Detailed Description

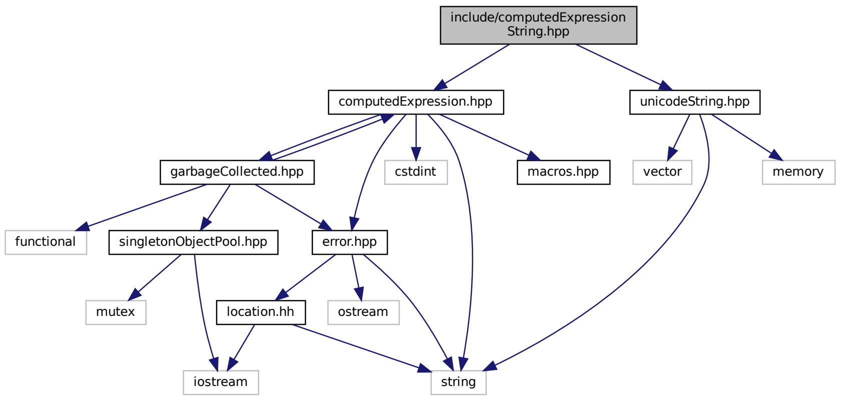
Declare the Tang::ComputedExpressionIteratorEnd class.

6.37 include/computedExpressionString.hpp File Reference

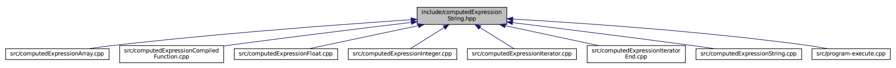
Declare the Tang::ComputedExpressionString class.

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

Include dependency graph for computedExpressionString.hpp:



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


include/computedExpression String.hpp
src/computedExpressionArray.cpp
src/computedExpressionCompiled Function.cpp
src/computedExpressionFloat.cpp
src/computedExpressionInteger.cpp
src/computedExpressionIterator.cpp
src/computedExpressionIterator End.cpp
src/computedExpressionString.cpp
src/program-execute.cpp

Classes

- class Tang::ComputedExpressionString

  Represents a String that is the result of a computation.

6.37.1 Detailed Description

Declare the Tang::ComputedExpressionString class.

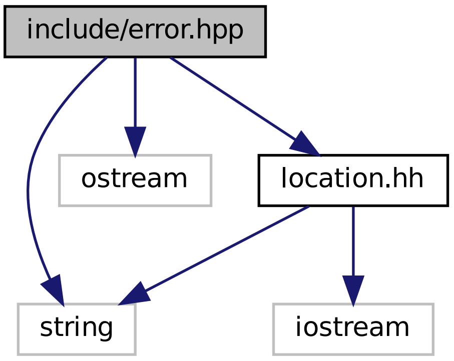
6.38 include/error.hpp File Reference

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

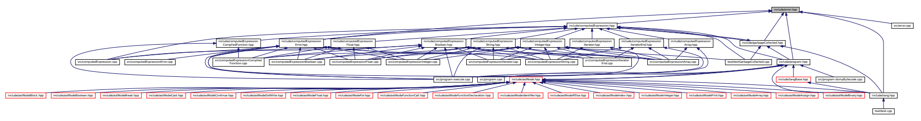
```
#include <string>
#include <ostream>
```

```
#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.

6.38.1 Detailed Description

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

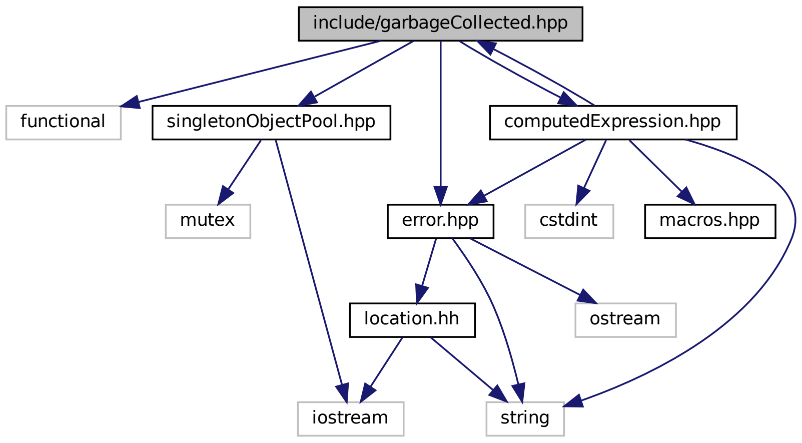
6.39 include/garbageCollected.hpp File Reference

Declare the Tang::GarbageCollected class.

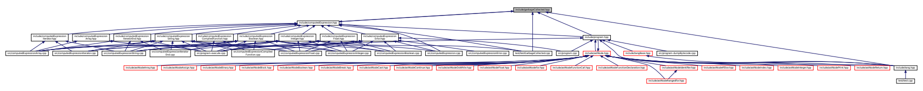
```
#include <functional>
#include "singletonObjectPool.hpp"
#include "computedExpression.hpp"
```

`#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.

6.39.1 Detailed Description

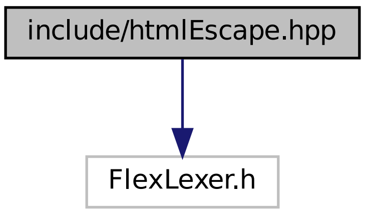
Declare the Tang::GarbageCollected class.

6.40 include/htmlEscape.hpp File Reference

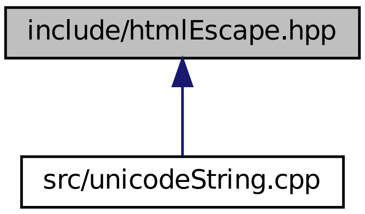
Declare the Tang::HtmlEscape used to tokenize a Tang script.

```
#include <FlexLexer.h>
```

Include dependency graph for htmlEscape.hpp:



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



Classes

- class Tang::HtmlEscape

  The Flex lexer class for the main Tang language.

Macros

- #define **yyFlexLexer** TangHtmlEscapeFlexLexer
- #define **YY_DECL** std::string Tang::HtmlEscape::get_next_token()

6.40.1 Detailed Description

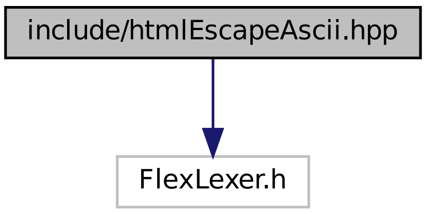
Declare the Tang::HtmlEscape used to tokenize a Tang script.

6.41 include/htmlEscapeAscii.hpp File Reference

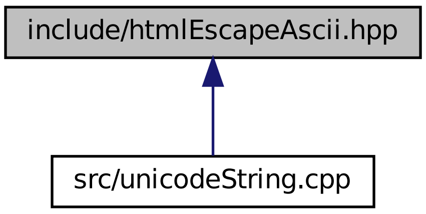
Declare the Tang::HtmlEscapeAscii used to tokenize a Tang script.

```
#include <FlexLexer.h>
```

Include dependency graph for htmlEscapeAscii.hpp:



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



Classes

- class Tang::HtmlEscapeAscii

  The Flex lexer class for the main Tang language.

Macros

- #define **yyFlexLexer** TangHtmlEscapeAsciiFlexLexer
- #define **YY_DECL** std::string Tang::HtmlEscapeAscii::get_next_token()

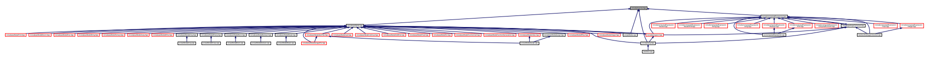
6.41.1 Detailed Description

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

6.42 include/macros.hpp File Reference

Contains generic macros.

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



Typedefs

- using Tang::integer_t = int32_t

  Define the size of signed integers used by Tang.
- using Tang::uinteger_t = int32_t

  Define the size of integers used by Tang.
- using Tang::float_t = float

  Define the size of floats used by Tang.

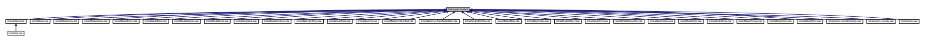
6.42.1 Detailed Description

Contains generic macros.

6.43 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 , COPY ,
  JMP , JMPF , JMPF_POP , JMPT ,
  JMPT_POP , NULLVAL , INTEGER , FLOAT ,
  BOOLEAN , STRING , ARRAY , FUNCTION ,
  ASSIGNINDEX , ADD , SUBTRACT , MULTIPLY ,
  DIVIDE , MODULO , NEGATIVE , NOT ,
  LT , LTE , GT , GTE ,
  EQ , NEQ , INDEX , SLICE ,
  GETITERATOR , ITERATORNEXT , ISITERATOREND , CASTINTEGER ,
  CASTFLOAT , CASTBOOLEAN , CALLFUNC , RETURN ,
  PRINT }

6.43.1 Detailed Description

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

6.43.2 Enumeration Type Documentation

6.43.2.1 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 #.
COPY	Stack # (from fp): Deep copy val @ stack #, 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.
ARRAY	Get len, pop `len` items, putting them into an array with the last array item popped first.
FUNCTION	Get argc, PC#: push function(argc, PC #)
ASSIGNINDEX	Pop index, pop collection, pop value, push (collection[index] = value)
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.
INDEX	Pop index, pop collection, push collection[index].
SLICE	Pop skip, pop end, pop begin, pop collection, push collection[begin:end:skip].
GETITERATOR	Pop a collection, push the collection iterator.

Enumerator

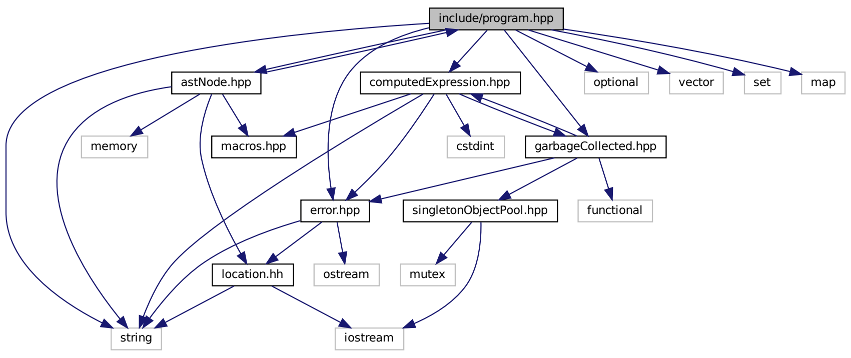
ITERATORNEXT	Pop an iterator, push the next iterator value.
ISITERATOREND	Pop a val, push bool(is val == iterator end)
CASTINTEGER	Pop a val, typecast to int, push.
CASTFLOAT	Pop a val, typecast to float, push.
CASTBOOLEAN	Pop a val, typecast to boolean, push.
CALLFUNC	Get argc, Pop a function, execute function if argc matches.
RETURN	Get stack #, pop return val, pop (stack #) times, push val, restore fp, restore pc.
PRINT	Pop val, print(val), push error or NULL.

6.44 include/program.hpp File Reference

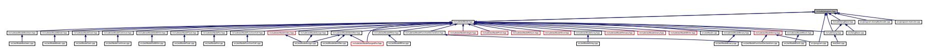
Declare the Tang::Program class used to compile and execute source code.

```
#include <string>
#include <optional>
#include <vector>
#include <set>
#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< Tang::uinteger_t >

  Contains the Opcodes of a compiled program.

6.44.1 Detailed Description

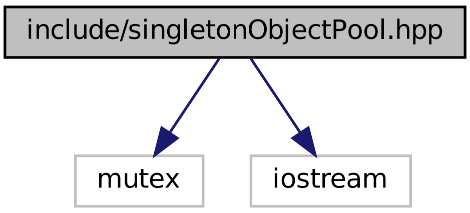
Declare the Tang::Program class used to compile and execute source code.

6.45 include/singletonObjectPool.hpp File Reference

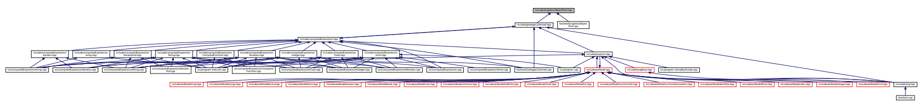
Declare the Tang::SingletonObjectPool class.

```
#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.

6.45.1 Detailed Description

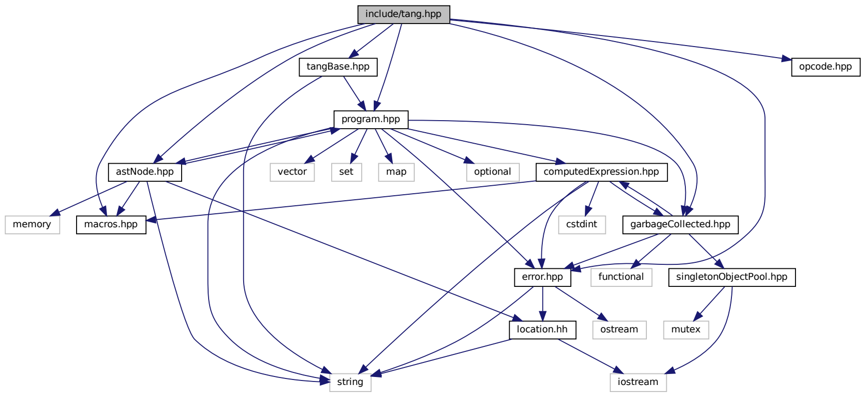
Declare the Tang::SingletonObjectPool class.

6.46 include/tang.hpp File Reference

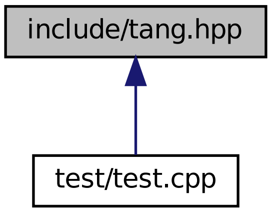
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

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

Include dependency graph for tang.hpp:



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



6.46.1 Detailed Description

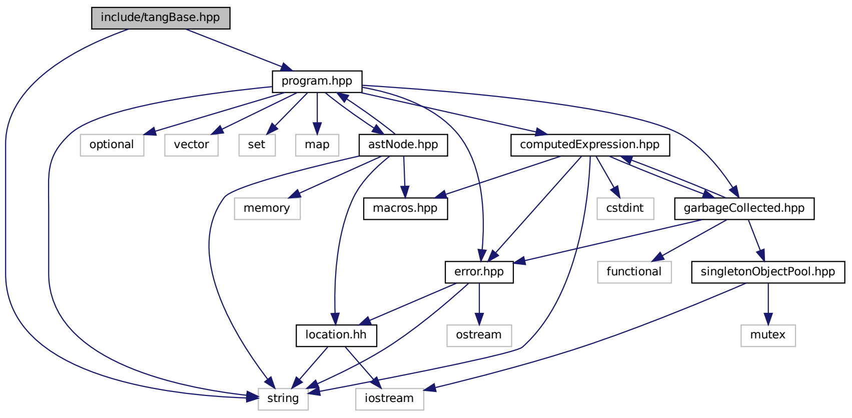
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

6.47 include/tangBase.hpp File Reference

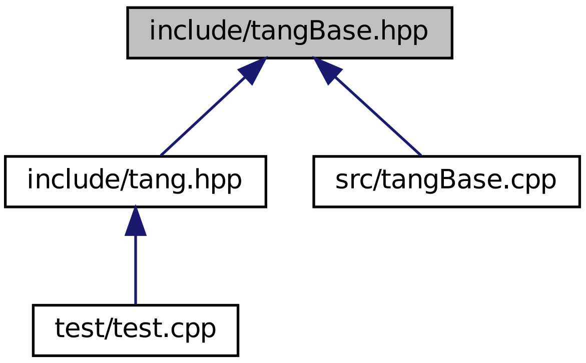
Declare the Tang::TangBase class used to interact with Tang.

```
#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.

6.47.1 Detailed Description

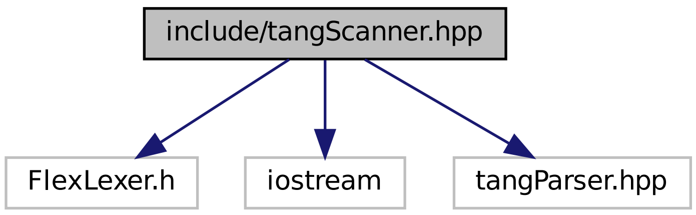
Declare the Tang::TangBase class used to interact with Tang.

6.48 include/tangScanner.hpp File Reference

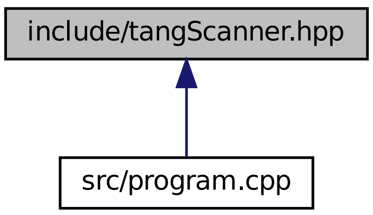
Declare the Tang::TangScanner used to tokenize a Tang script.

```
#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()

6.48.1 Detailed Description

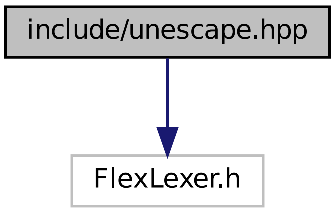
Declare the Tang::TangScanner used to tokenize a Tang script.

6.49 include/unescape.hpp File Reference

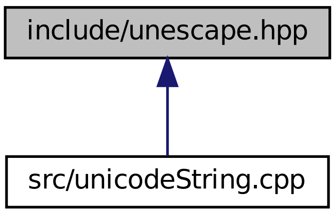
Declare the Tang::Unescape used to tokenize a Tang script.

```
#include <FlexLexer.h>
```

Include dependency graph for unescape.hpp:



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



Classes

- class Tang::Unescape

  The Flex lexer class for the main Tang language.

Macros

- #define **yyFlexLexer** TangUnescapeFlexLexer
- #define **YY_DECL** std::string Tang::Unescape::get_next_token()

6.49.1 Detailed Description

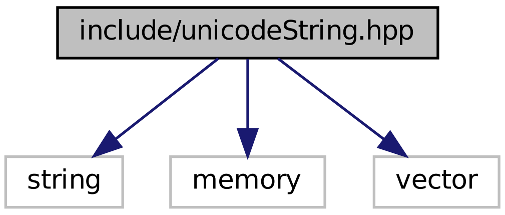
Declare the Tang::Unescape used to tokenize a Tang script.

6.50 include/unicodeString.hpp File Reference

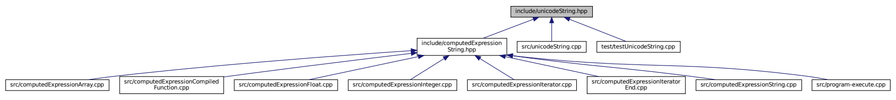
Contains the code to interface with the ICU library.

```
#include <string>
#include <memory>
#include <vector>
```

Include dependency graph for unicodeString.hpp:



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


include/unicodeString.hpp
include/computedExpression String.hpp
src/unicodeString.cpp
test/testUnicodeString.cpp
src/computedExpressionArray.cpp
src/computedExpressionCompiled Function.cpp
src/computedExpressionFloat.cpp
src/computedExpressionInteger.cpp
src/computedExpressionIterator.cpp
src/computedExpressionIterator End.cpp
src/computedExpressionString.cpp
src/program-execute.cpp

Classes

- class Tang::UnicodeString

  Represents a UTF-8 encoded string that is Unicode-aware.

Functions

- std::string Tang::unescape (const std::string &str)

  Return an "unescaped" version of the provided string, which, when interpreted by Tang, should result in a representation equivalent to the original source string.
- std::string Tang::htmlEscape (const std::string &str)

  Return an "html escaped" version of the provided string.
- std::string Tang::htmlEscapeAscii (const std::string &str)

  Return an Ascii-only, "html escaped" version of the provided string.

6.50.1 Detailed Description

Contains the code to interface with the ICU library.

6.50.2 Function Documentation

6.50.2.1 htmlEscape()

```
string Tang::htmlEscape (
            const std::string & str )
```

Return an "html escaped" version of the provided string.

Only "critical" characters $<$, $>$, `&`, `"`, and ''` will be escaped. All other characters will be allowed through unaltered. The result is a UTF-8 encoded string that is safe for inclusion in an HTML template without disturbing the HTML structure.

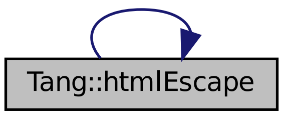
Parameters

str	The string to be escaped.

Returns

An "escaped" version of the provided string.

Here is the call graph for this function:



6.50.2.2 htmlEscapeAscii()

```
string Tang::htmlEscapeAscii (
            const std::string & str )
```

Return an Ascii-only, "html escaped" version of the provided string.

This function will convert all characters into an Ascii-only representation of the provided UTF-8 encoded string. Visible, standard Ascii characters will pass through unaltered, but all others will be replaced by their HTML escape sequence (if it exists), or the appropriate hexadecimal escape code.

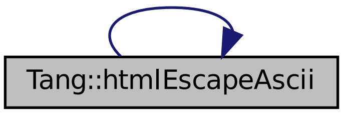
Parameters

str	The string to be escaped.

Returns

An "escaped" version of the provided string.

Here is the call graph for this function:



6.50.2.3 unescape()

```
string Tang::unescape (
            const std::string & str )
```

Return an "unescaped" version of the provided string, which, when interpreted by Tang, should result in a representation equivalent to the original source string.

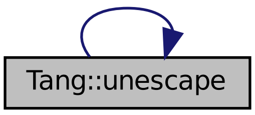
Parameters

str	The string to be unescaped.

Returns

An "unescaped" version of the provided string.

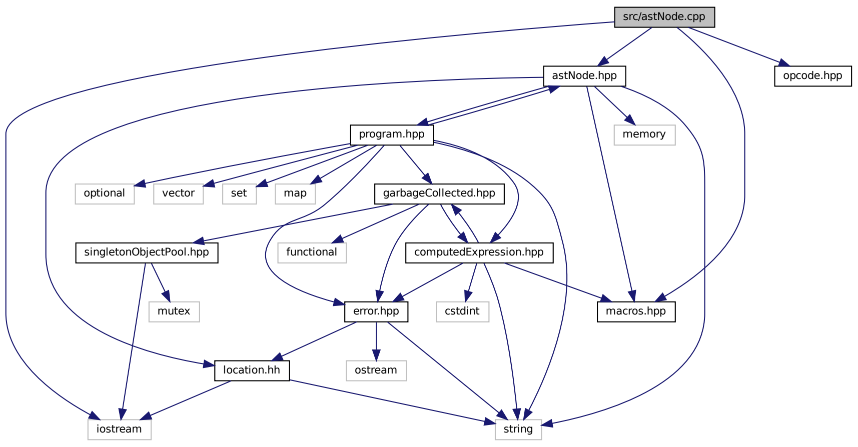
Here is the call graph for this function:



6.51 src/astNode.cpp File Reference

Define the Tang::AstNode class.

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



6.51.1 Detailed Description

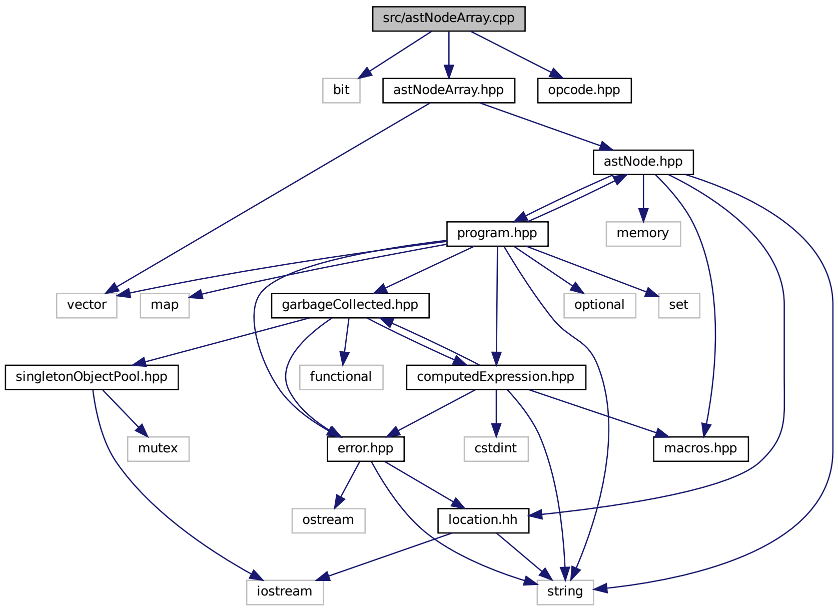
Define the Tang::AstNode class.

6.52 src/astNodeArray.cpp File Reference

Define the Tang::AstNodeArray class.

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

Include dependency graph for astNodeArray.cpp:



6.52.1 Detailed Description

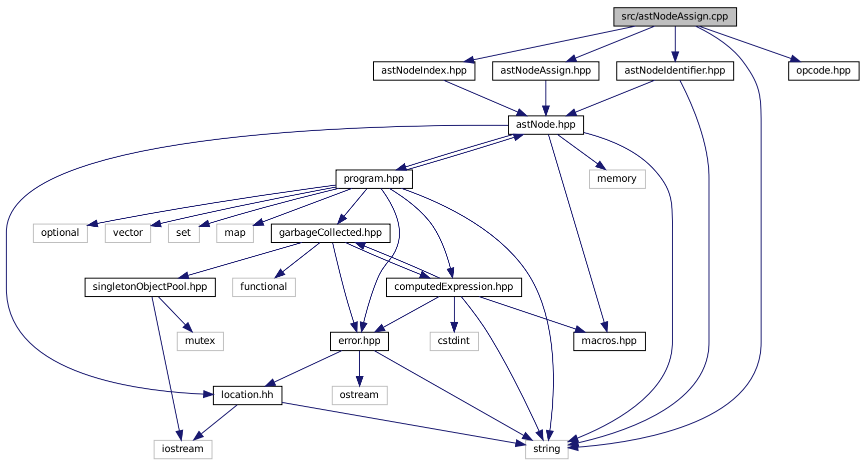
Define the Tang::AstNodeArray class.

6.53 src/astNodeAssign.cpp File Reference

Define the Tang::AstNodeAssign class.

```
#include <string>
#include "astNodeAssign.hpp"
#include "astNodeIdentifier.hpp"
#include "astNodeIndex.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeAssign.cpp:



6.53.1 Detailed Description

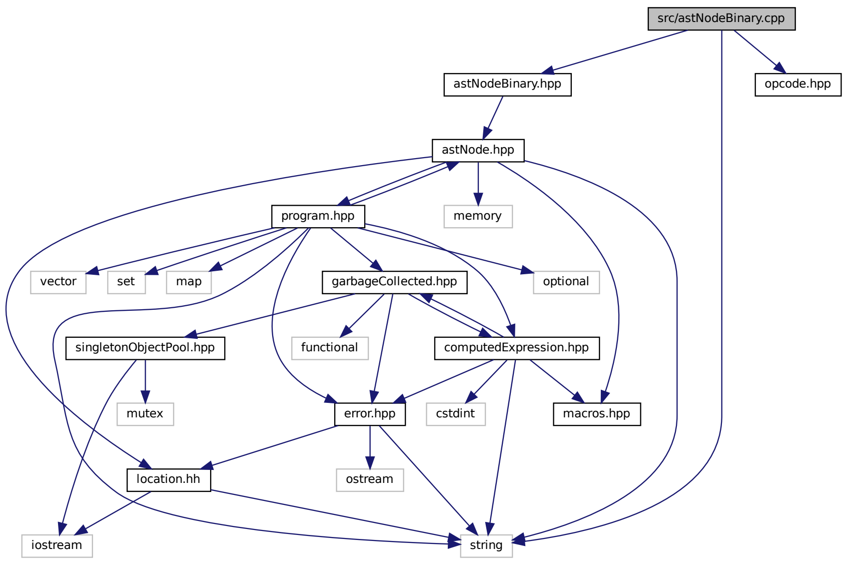
Define the Tang::AstNodeAssign class.

6.54 src/astNodeBinary.cpp File Reference

Define the Tang::AstNodeBinary class.

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

Include dependency graph for astNodeBinary.cpp:



6.54.1 Detailed Description

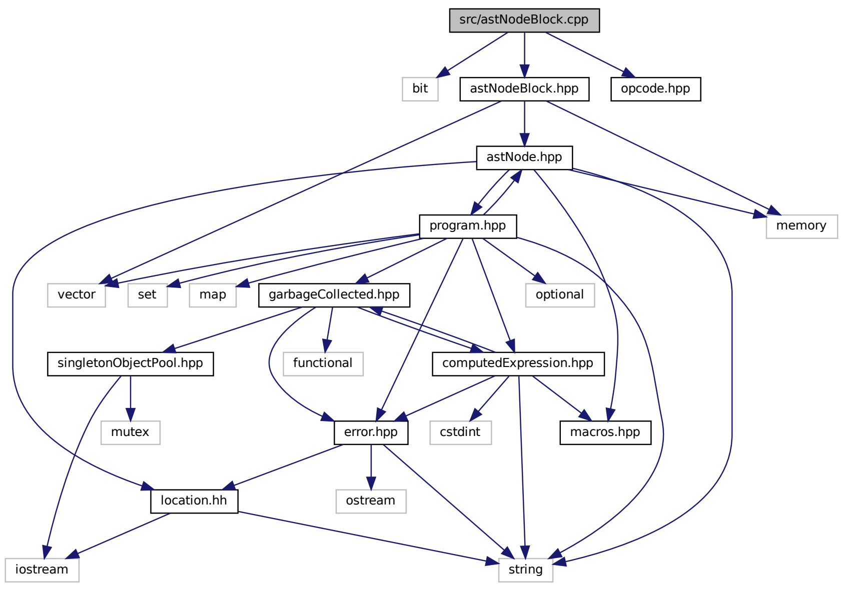
Define the Tang::AstNodeBinary class.

6.55 src/astNodeBlock.cpp File Reference

Define the Tang::AstNodeBlock class.

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

Include dependency graph for astNodeBlock.cpp:



6.55.1 Detailed Description

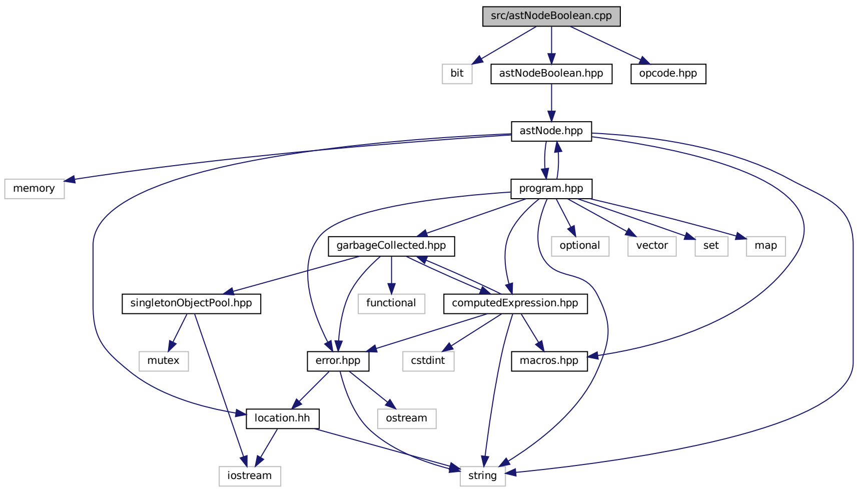
Define the Tang::AstNodeBlock class.

6.56 src/astNodeBoolean.cpp File Reference

Define the Tang::AstNodeBoolean class.

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

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



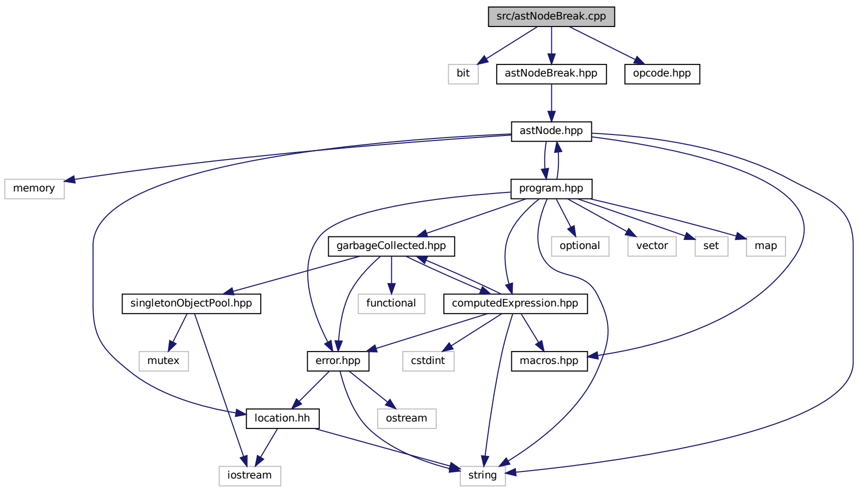
6.56.1 Detailed Description

Define the Tang::AstNodeBoolean class.

6.57 src/astNodeBreak.cpp File Reference

Define the Tang::AstNodeBreak class.

```
#include <bit>
#include "astNodeBreak.hpp"
#include "opcode.hpp"
```
Include dependency graph for astNodeBreak.cpp:



6.57.1 Detailed Description

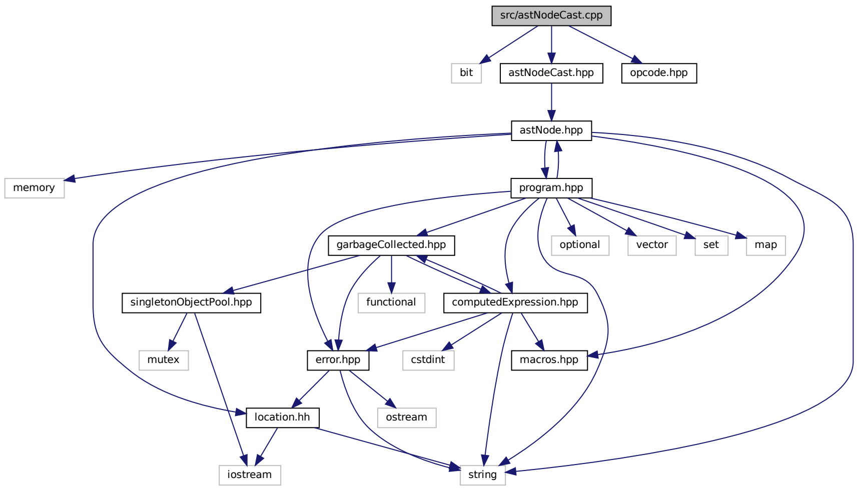
Define the Tang::AstNodeBreak class.

6.58 src/astNodeCast.cpp File Reference

Define the Tang::AstNodeCast class.

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

Include dependency graph for astNodeCast.cpp:



6.58.1 Detailed Description

Define the Tang::AstNodeCast class.

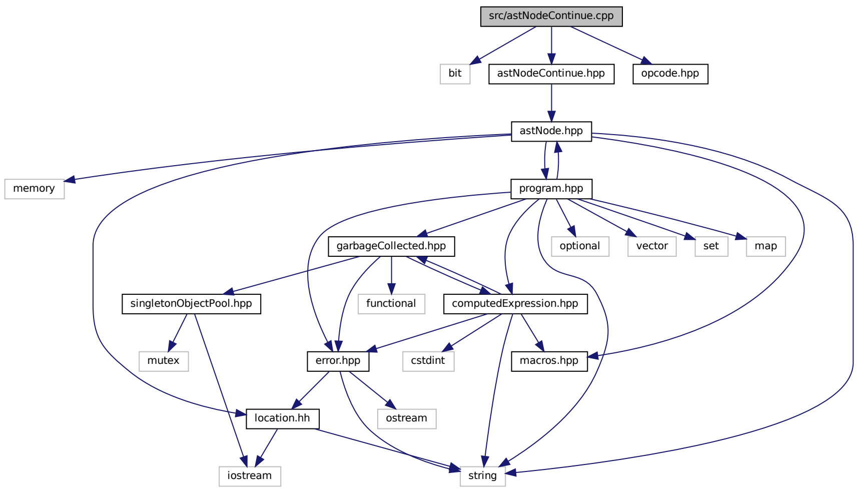
6.59 src/astNodeContinue.cpp File Reference

Define the Tang::AstNodeContinue class.

```
#include <bit>
#include "astNodeContinue.hpp"
```

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

Include dependency graph for astNodeContinue.cpp:



6.59.1 Detailed Description

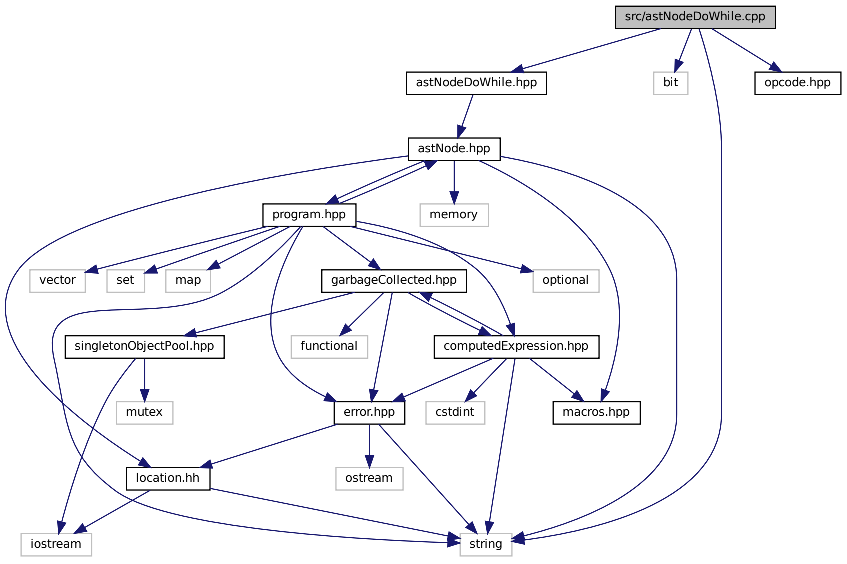
Define the Tang::AstNodeContinue class.

6.60 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:



6.60.1 Detailed Description

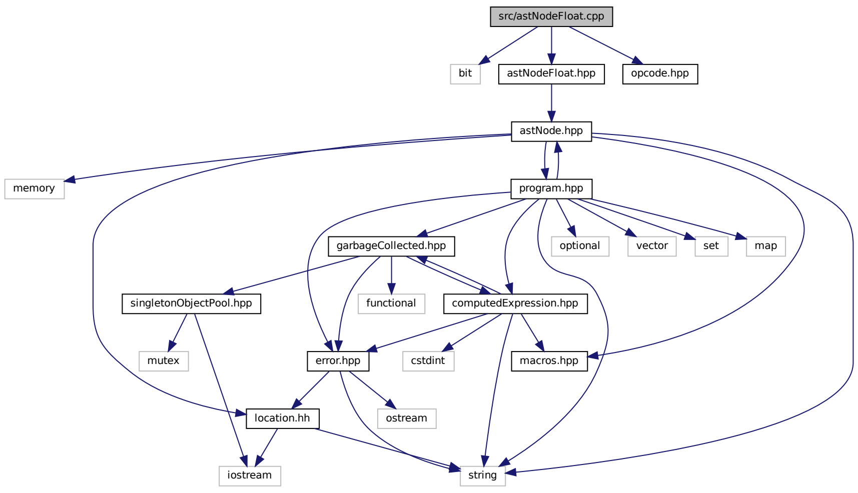
Define the Tang::AstNodeDoWhile class.

6.61 src/astNodeFloat.cpp File Reference

Define the Tang::AstNodeFloat class.

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

Include dependency graph for astNodeFloat.cpp:



6.61.1 Detailed Description

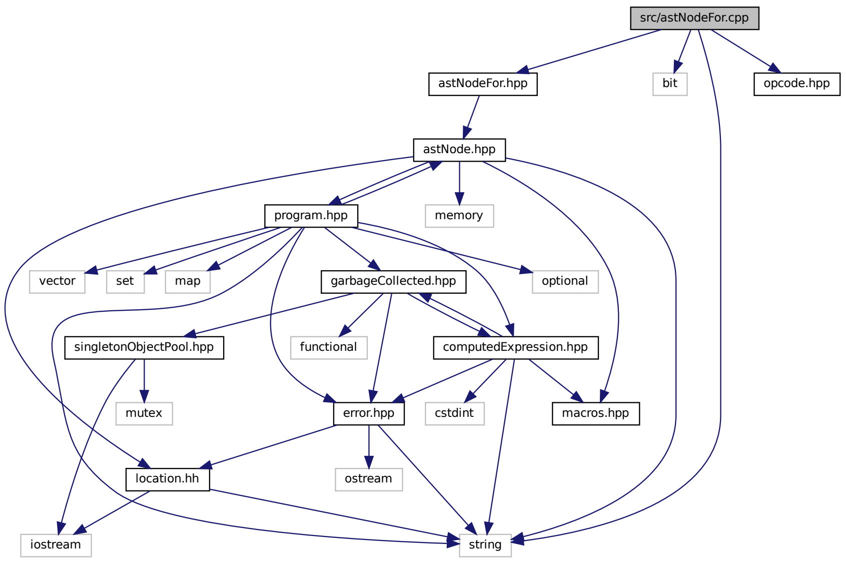
Define the Tang::AstNodeFloat class.

6.62 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:



6.62.1 Detailed Description

Define the Tang::AstNodeFor class.

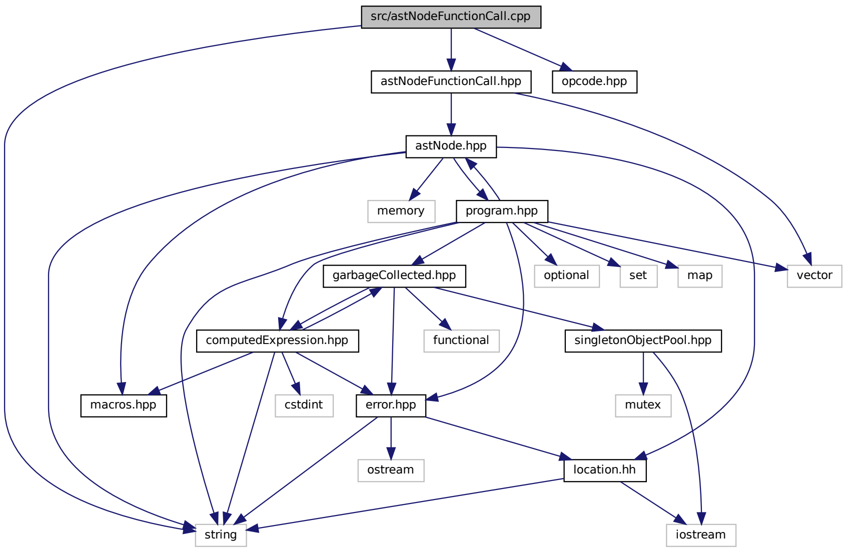
6.63 src/astNodeFunctionCall.cpp File Reference

Define the Tang::AstNodeFunctionCall class.

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

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

Include dependency graph for astNodeFunctionCall.cpp:



6.63.1 Detailed Description

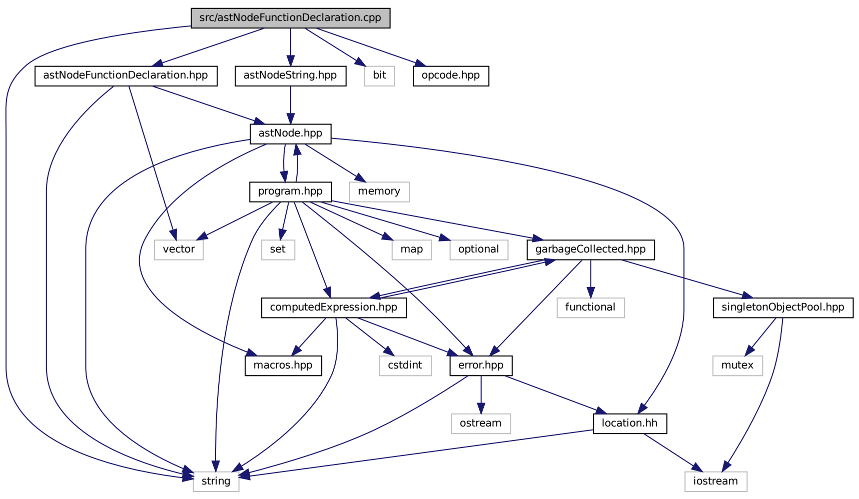
Define the Tang::AstNodeFunctionCall class.

6.64 src/astNodeFunctionDeclaration.cpp File Reference

Define the Tang::AstNodeFunctionDeclaration class.

```
#include <string>
#include <bit>
#include "astNodeFunctionDeclaration.hpp"
#include "astNodeString.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeFunctionDeclaration.cpp:



6.64.1 Detailed Description

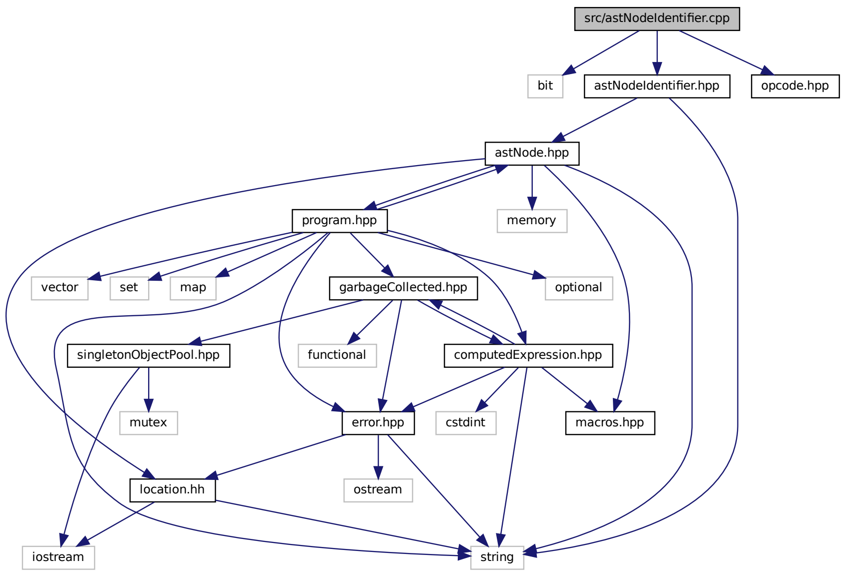
Define the Tang::AstNodeFunctionDeclaration class.

6.65 src/astNodeIdentifier.cpp File Reference

Define the Tang::AstNodeIdentifier class.

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

Include dependency graph for astNodeIdentifier.cpp:



6.65.1 Detailed Description

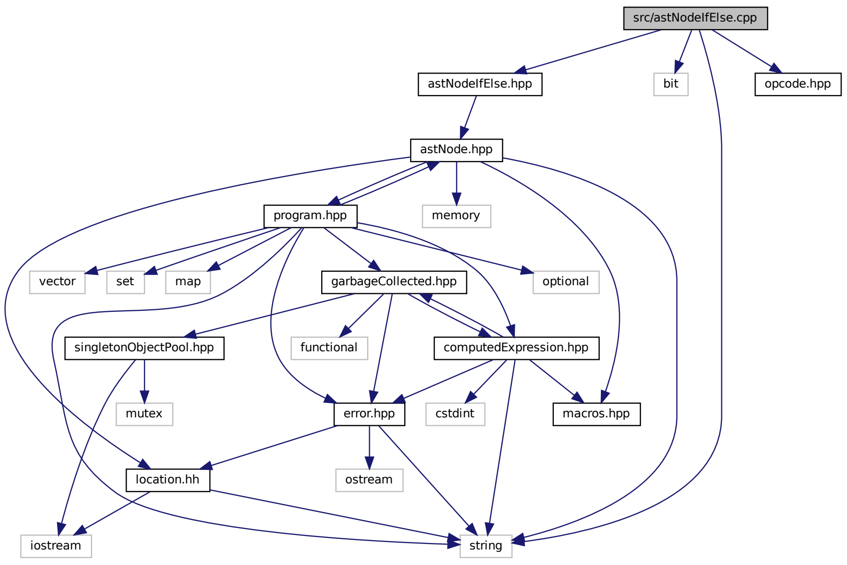
Define the Tang::AstNodeIdentifier class.

6.66 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:



6.66.1 Detailed Description

Define the Tang::AstNodeIfElse class.

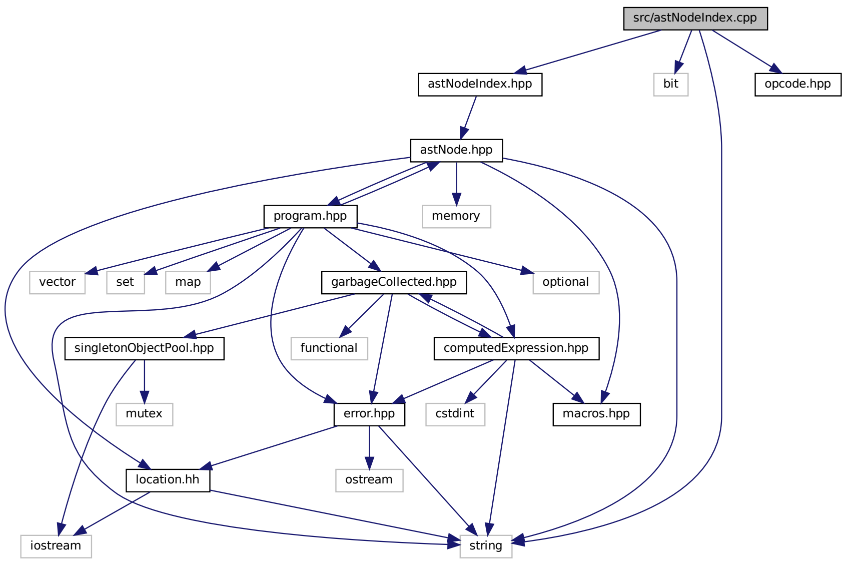
6.67 src/astNodeIndex.cpp File Reference

Define the Tang::AstNodeIndex class.

```
#include <string>
#include <bit>
#include "astNodeIndex.hpp"
```

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

Include dependency graph for astNodeIndex.cpp:



6.67.1 Detailed Description

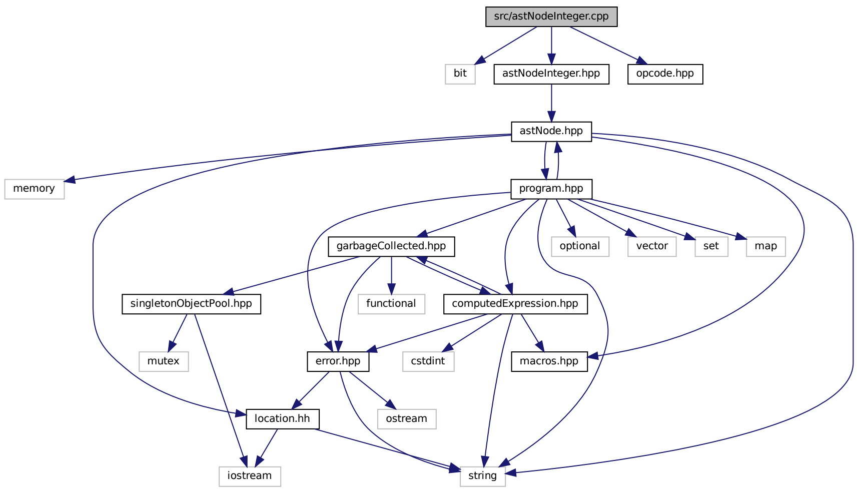
Define the Tang::AstNodeIndex class.

6.68 src/astNodeInteger.cpp File Reference

Define the Tang::AstNodeInteger class.

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

Include dependency graph for astNodeInteger.cpp:



6.68.1 Detailed Description

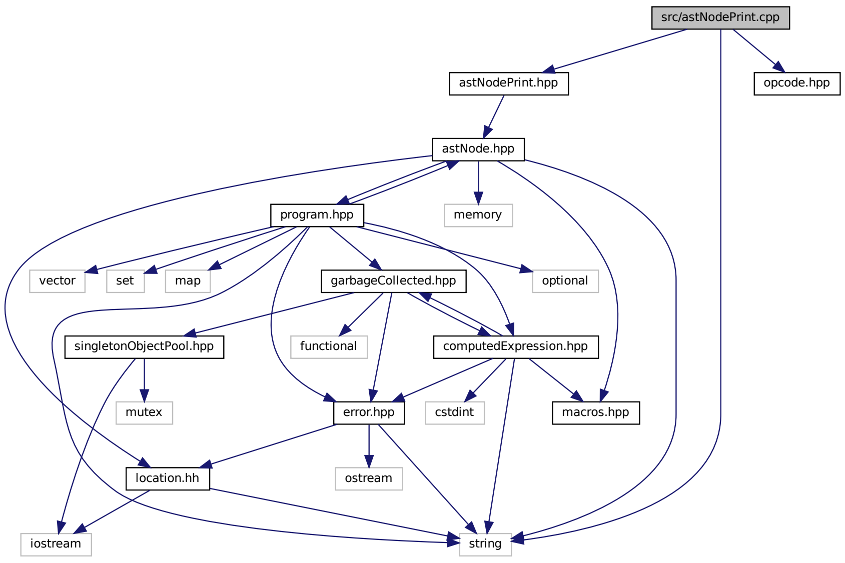
Define the Tang::AstNodeInteger class.

6.69 src/astNodePrint.cpp File Reference

Define the Tang::AstNodePrint class.

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

Include dependency graph for astNodePrint.cpp:



6.69.1 Detailed Description

Define the Tang::AstNodePrint class.

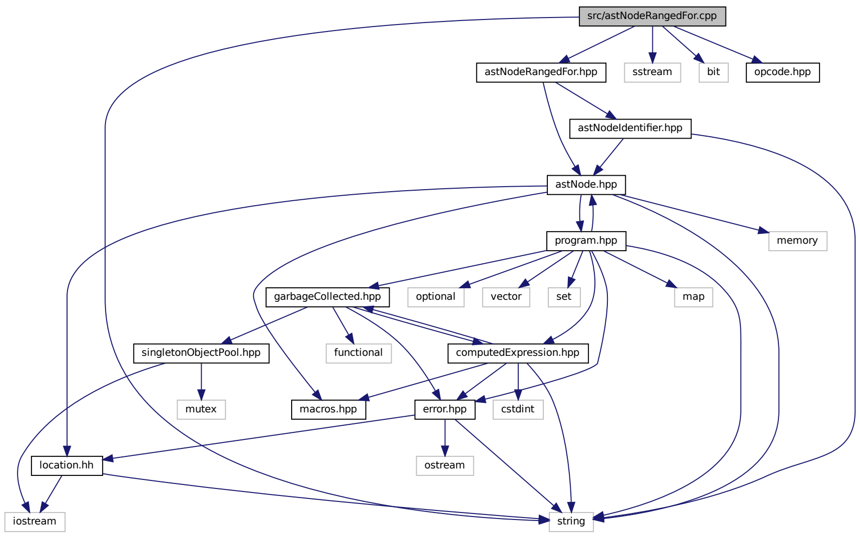
6.70 src/astNodeRangedFor.cpp File Reference

Define the Tang::AstNodeRangedFor class.

```
#include <string>
#include <sstream>
#include <bit>
#include "astNodeRangedFor.hpp"
```

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

Include dependency graph for astNodeRangedFor.cpp:



6.70.1 Detailed Description

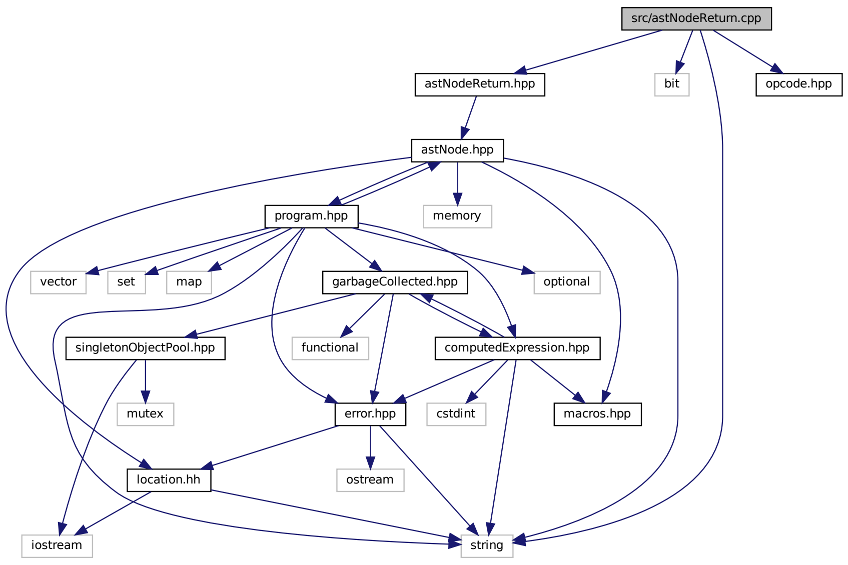
Define the Tang::AstNodeRangedFor class.

6.71 src/astNodeReturn.cpp File Reference

Define the Tang::AstNodeReturn class.

```
#include <string>
#include <bit>
#include "astNodeReturn.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeReturn.cpp:



6.71.1 Detailed Description

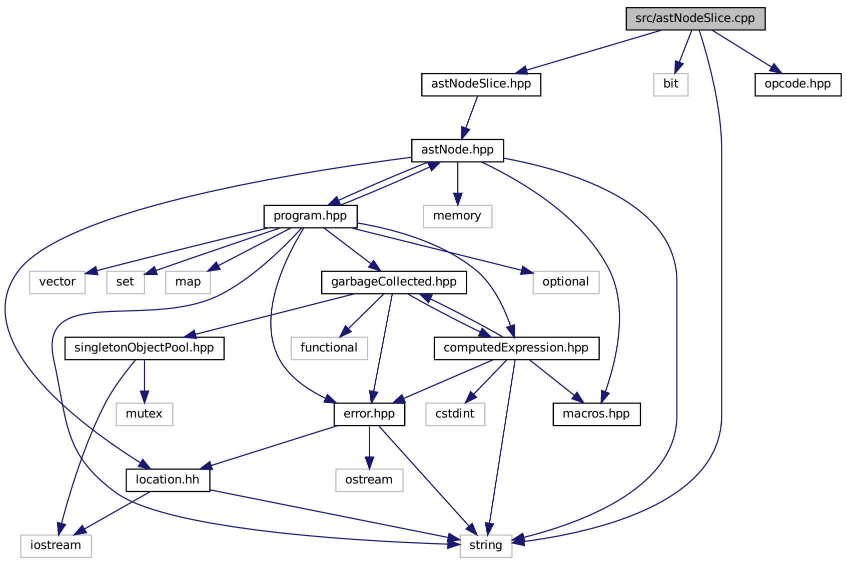
Define the Tang::AstNodeReturn class.

6.72 src/astNodeSlice.cpp File Reference

Define the Tang::AstNodeSlice class.

```
#include <string>
#include <bit>
#include "astNodeSlice.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeSlice.cpp:



6.72.1 Detailed Description

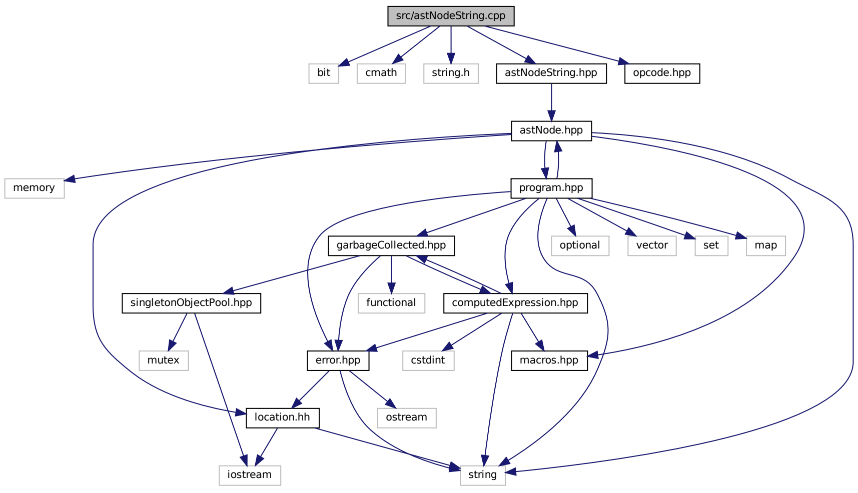
Define the Tang::AstNodeSlice class.

6.73 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:



6.73.1 Detailed Description

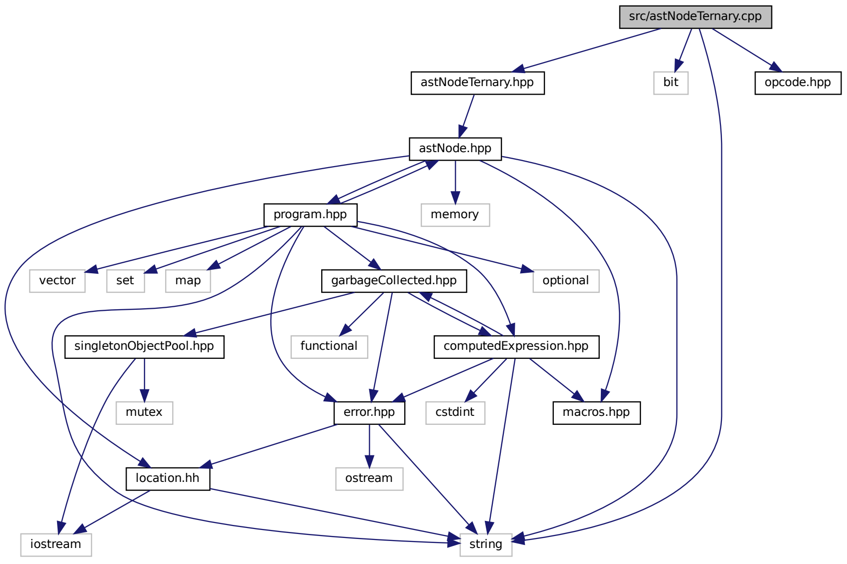
Define the Tang::AstNodeString class.

6.74 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:



6.74.1 Detailed Description

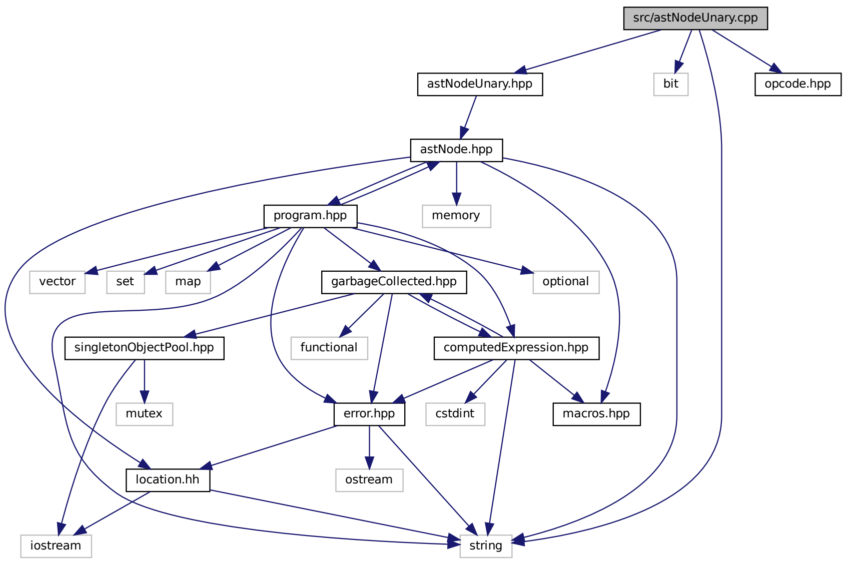
Define the Tang::AstNodeTernary class.

6.75 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:



6.75.1 Detailed Description

Define the Tang::AstNodeUnary class.

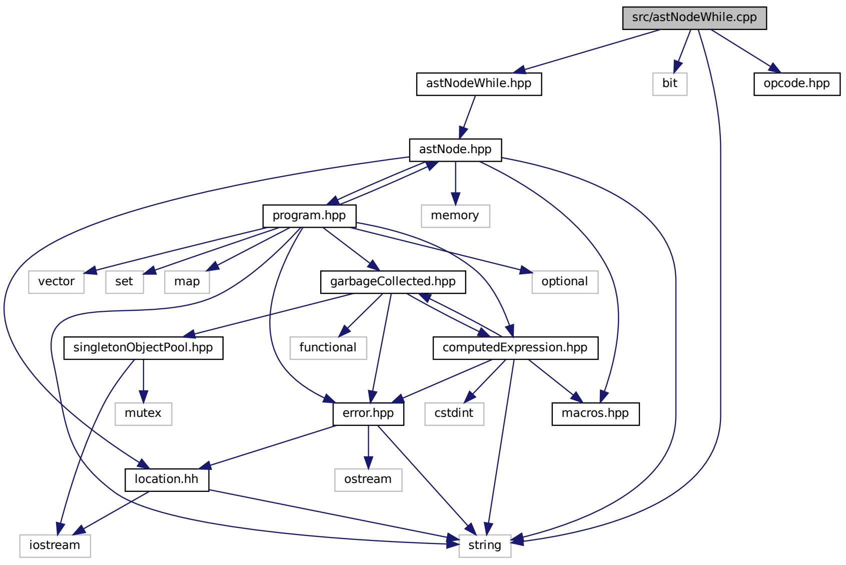
6.76 src/astNodeWhile.cpp File Reference

Define the Tang::AstNodeWhile class.

```
#include <string>
#include <bit>
#include "astNodeWhile.hpp"
```

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

Include dependency graph for astNodeWhile.cpp:



6.76.1 Detailed Description

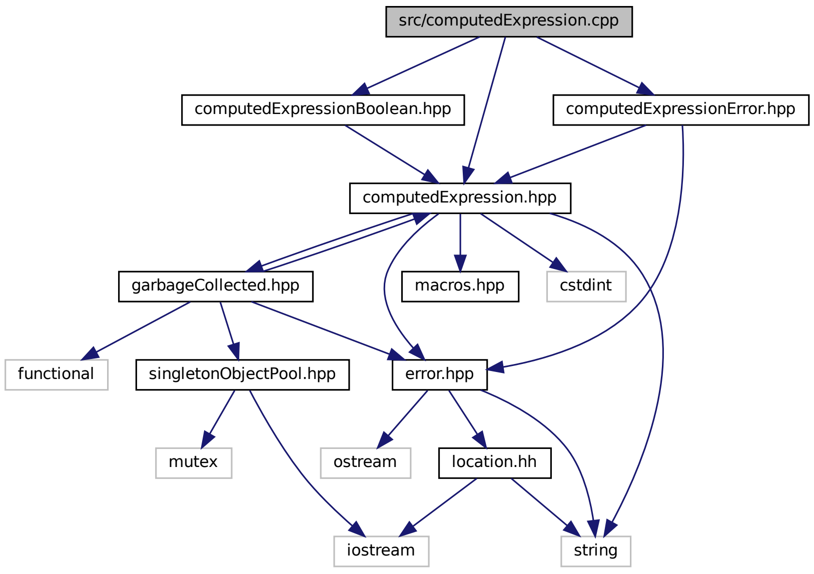
Define the Tang::AstNodeWhile class.

6.77 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:



6.77.1 Detailed Description

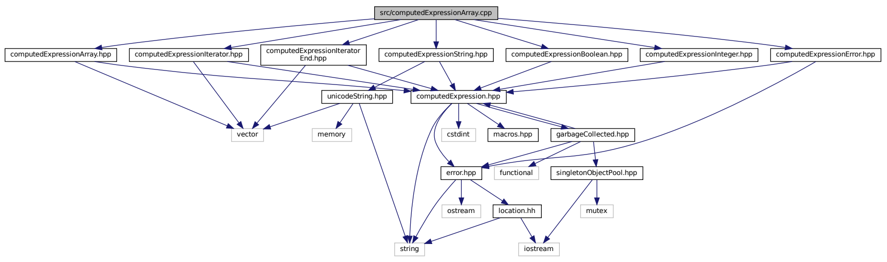
Define the Tang::ComputedExpression class.

6.78 src/computedExpressionArray.cpp File Reference

Define the Tang::ComputedExpressionArray class.

```
#include "computedExpressionArray.hpp"
#include "computedExpressionInteger.hpp"
#include "computedExpressionBoolean.hpp"
#include "computedExpressionString.hpp"
#include "computedExpressionIterator.hpp"
#include "computedExpressionIteratorEnd.hpp"
#include "computedExpressionError.hpp"
```

Include dependency graph for computedExpressionArray.cpp:



6.78.1 Detailed Description

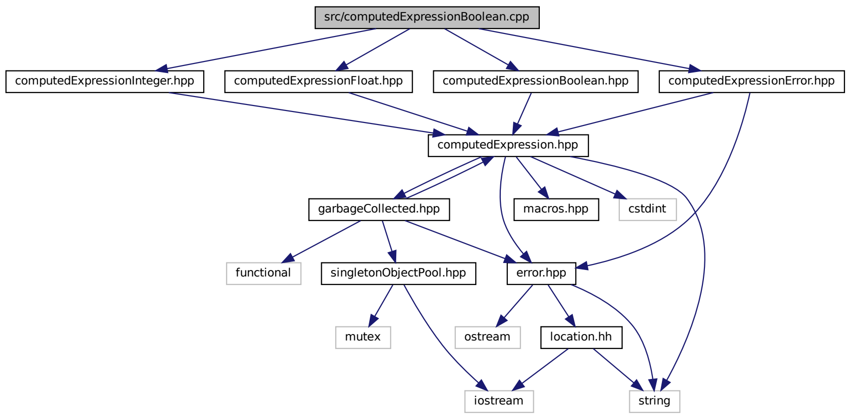
Define the Tang::ComputedExpressionArray class.

6.79 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
computedExpressionInteger.hpp
computedExpressionFloat.hpp
computedExpressionBoolean.hpp
computedExpressionError.hpp
computedExpression.hpp
garbageCollected.hpp
macros.hpp
cstdint
functional
singletonObjectPool.hpp
error.hpp
mutex
ostream
location.hh
iostream
string

6.79.1 Detailed Description

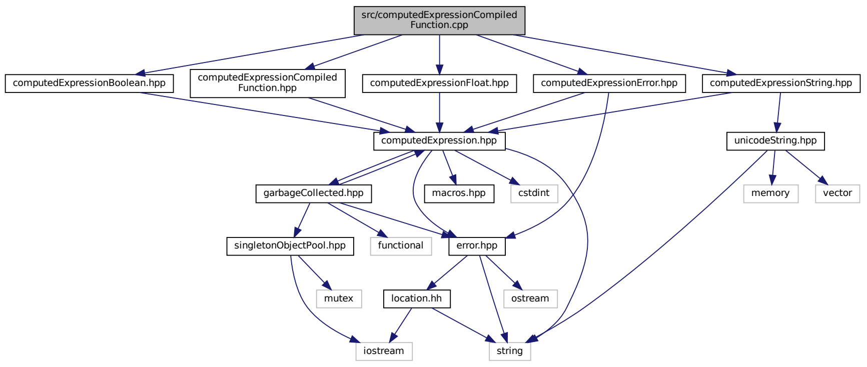
Define the Tang::ComputedExpressionBoolean class.

6.80 src/computedExpressionCompiledFunction.cpp File Reference

Define the Tang::ComputedExpressionCompiledFunction class.

```
#include "computedExpressionCompiledFunction.hpp"
#include "computedExpressionFloat.hpp"
#include "computedExpressionBoolean.hpp"
#include "computedExpressionString.hpp"
```

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


src/computedExpressionCompiledFunction.cpp
computedExpressionBoolean.hpp
computedExpressionCompiledFunction.hpp
computedExpressionFloat.hpp
computedExpressionError.hpp
computedExpressionString.hpp
computedExpression.hpp
unicodeString.hpp
garbageCollected.hpp
macros.hpp
cstdint
memory
vector
singletonObjectPool.hpp
functional
error.hpp
mutex
location.hh
ostream
iostream
string

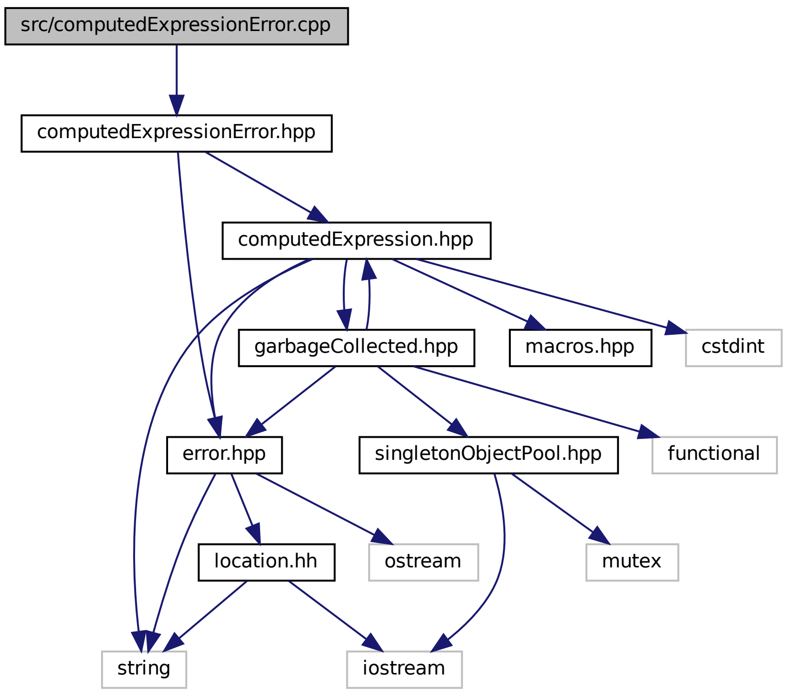
6.80.1 Detailed Description

Define the Tang::ComputedExpressionCompiledFunction class.

6.81 src/computedExpressionError.cpp File Reference

Define the Tang::ComputedExpressionError class.

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



6.81.1 Detailed Description

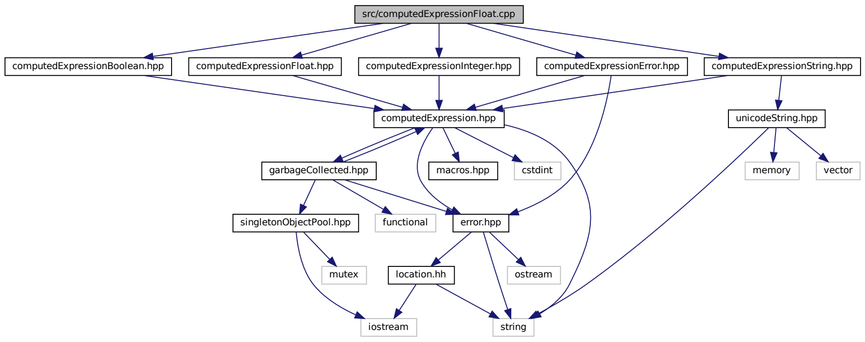
Define the Tang::ComputedExpressionError class.

6.82 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
computedExpressionBoolean.hpp
computedExpressionFloat.hpp
computedExpressionInteger.hpp
computedExpressionError.hpp
computedExpressionString.hpp
computedExpression.hpp
unicodeString.hpp
garbageCollected.hpp
macros.hpp
cstdint
memory
vector
singletonObjectPool.hpp
functional
error.hpp
mutex
location.hh
ostream
iostream
string

6.82.1 Detailed Description

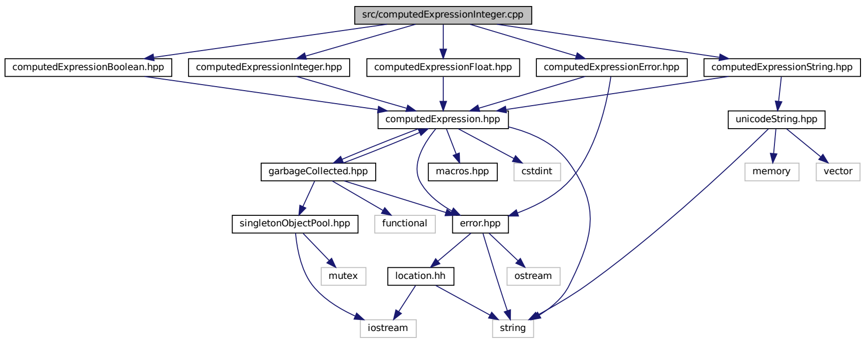
Define the Tang::ComputedExpressionFloat class.

6.83 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
computedExpressionBoolean.hpp
computedExpressionInteger.hpp
computedExpressionFloat.hpp
computedExpressionError.hpp
computedExpressionString.hpp
computedExpression.hpp
unicodeString.hpp
garbageCollected.hpp
macros.hpp
cstdint
memory
vector
singletonObjectPool.hpp
functional
error.hpp
mutex
location.hh
ostream
iostream
string

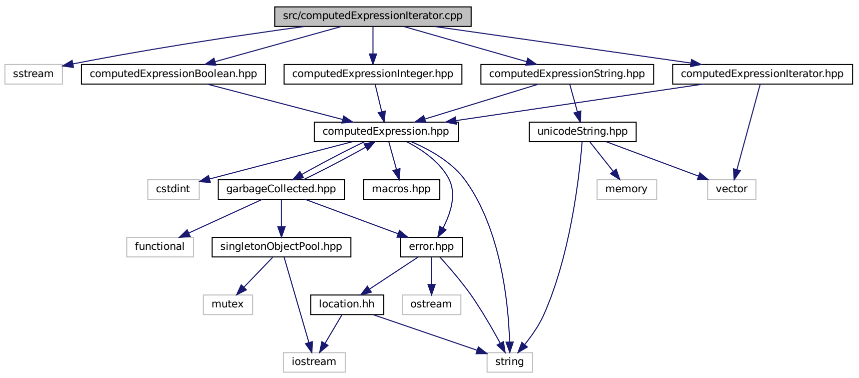
6.83.1 Detailed Description

Define the Tang::ComputedExpressionInteger class.

6.84 src/computedExpressionIterator.cpp File Reference

Define the Tang::ComputedExpressionIterator class.

```
#include <sstream>
#include "computedExpressionIterator.hpp"
#include "computedExpressionInteger.hpp"
#include "computedExpressionBoolean.hpp"
#include "computedExpressionString.hpp"
```
Include dependency graph for computedExpressionIterator.cpp:



6.84.1 Detailed Description

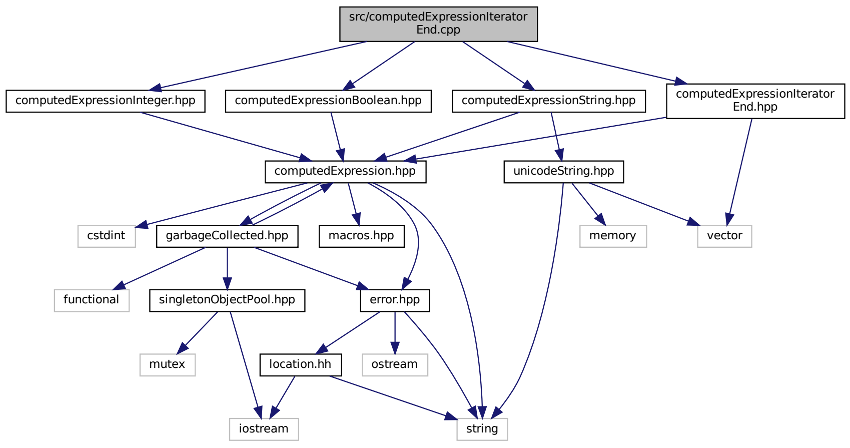
Define the Tang::ComputedExpressionIterator class.

6.85 src/computedExpressionIteratorEnd.cpp File Reference

Define the Tang::ComputedExpressionIteratorEnd class.

```
#include "computedExpressionIteratorEnd.hpp"
#include "computedExpressionInteger.hpp"
#include "computedExpressionBoolean.hpp"
#include "computedExpressionString.hpp"
```

Include dependency graph for computedExpressionIteratorEnd.cpp:



6.85.1 Detailed Description

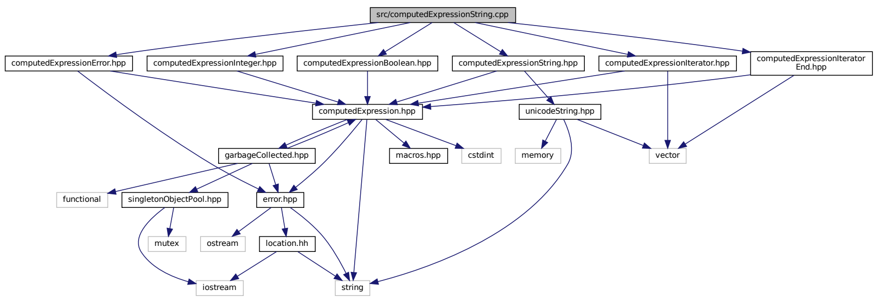
Define the Tang::ComputedExpressionIteratorEnd class.

6.86 src/computedExpressionString.cpp File Reference

Define the Tang::ComputedExpressionString class.

```
#include "computedExpressionString.hpp"
#include "computedExpressionBoolean.hpp"
#include "computedExpressionError.hpp"
#include "computedExpressionInteger.hpp"
#include "computedExpressionIterator.hpp"
#include "computedExpressionIteratorEnd.hpp"
```

Include dependency graph for computedExpressionString.cpp:


src/computedExpressionString.cpp
computedExpressionError.hpp
computedExpressionInteger.hpp
computedExpressionBoolean.hpp
computedExpressionString.hpp
computedExpressionIterator.hpp
computedExpressionIterator End.hpp
computedExpression.hpp
unicodeString.hpp
garbageCollected.hpp
macros.hpp
cstdint
memory
vector
functional
singletonObjectPool.hpp
error.hpp
mutex
ostream
location.hh
iostream
string

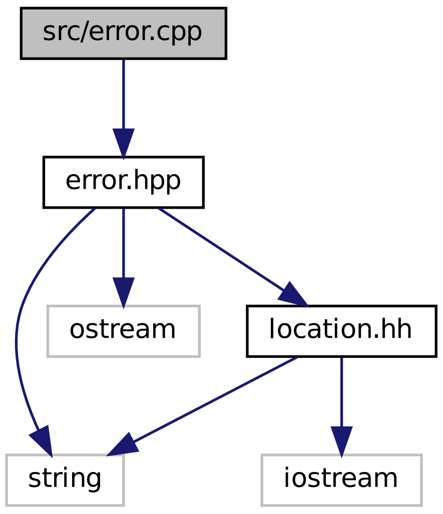
6.86.1 Detailed Description

Define the Tang::ComputedExpressionString class.

6.87 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)

6.87.1 Detailed Description

Define the Tang::Error class.

6.87.2 Function Documentation

6.87.2.1 operator<<()

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

Parameters

out	The output stream.
error	The Error object.

Returns

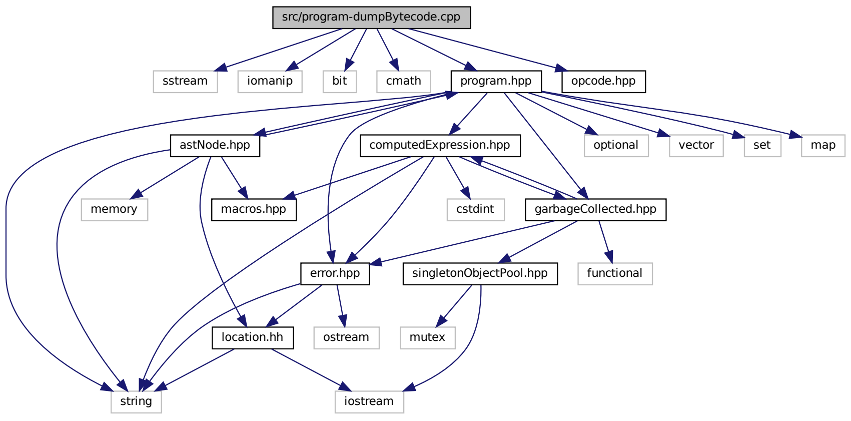
The output stream.

6.88 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.

6.88.1 Detailed Description

Define the Tang::Program::dumpBytecode method.

6.88.2 Macro Definition Documentation

6.88.2.1 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

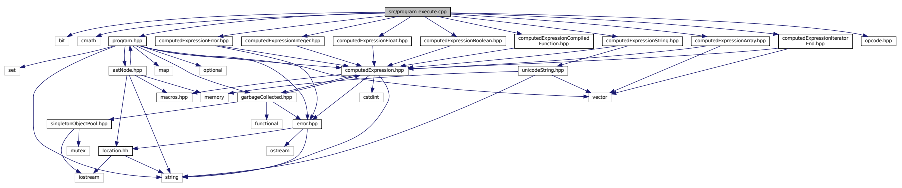
x	The number of additional vector entries that should exist.

6.89 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 "computedExpressionArray.hpp"
#include "computedExpressionCompiledFunction.hpp"
#include "computedExpressionIteratorEnd.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.

6.89.1 Detailed Description

Define the Tang::Program::execute method.

6.89.2 Macro Definition Documentation

6.89.2.1 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.

6.89.2.2 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

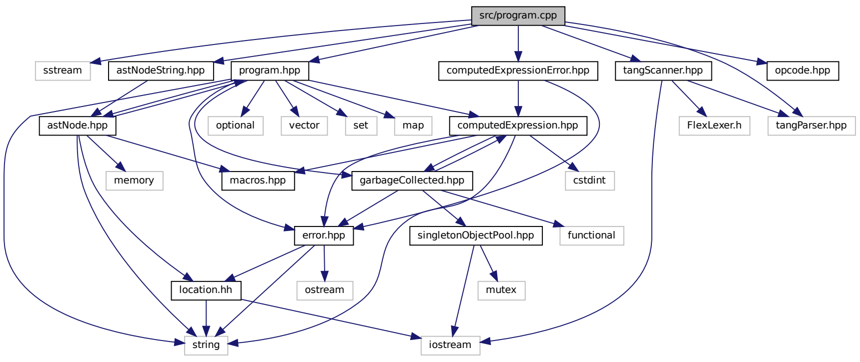
x	The number of entries that should exist in the stack.

6.90 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:



6.90.1 Detailed Description

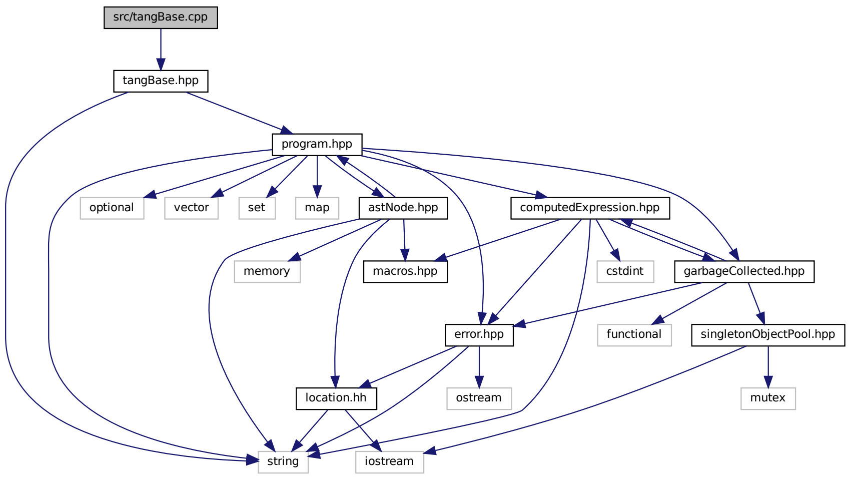
Define the Tang::Program class.

6.91 src/tangBase.cpp File Reference

Define the Tang::TangBase class.

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

Include dependency graph for tangBase.cpp:



6.91.1 Detailed Description

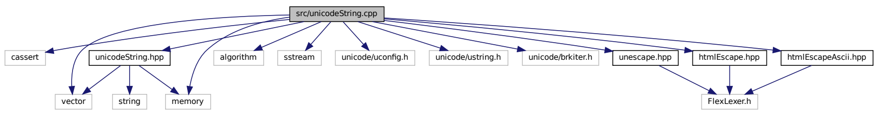
Define the Tang::TangBase class.

6.92 src/unicodeString.cpp File Reference

Contains the function declarations for the Tang::UnicodeString class and the interface to ICU.

```
#include <cassert>
#include <vector>
#include <memory>
#include <algorithm>
#include <sstream>
#include <unicode/uconfig.h>
#include <unicode/ustring.h>
#include <unicode/brkiter.h>
#include "unicodeString.hpp"
#include "unescape.hpp"
#include "htmlEscape.hpp"
#include "htmlEscapeAscii.hpp"
```

Include dependency graph for unicodeString.cpp:



6.92.1 Detailed Description

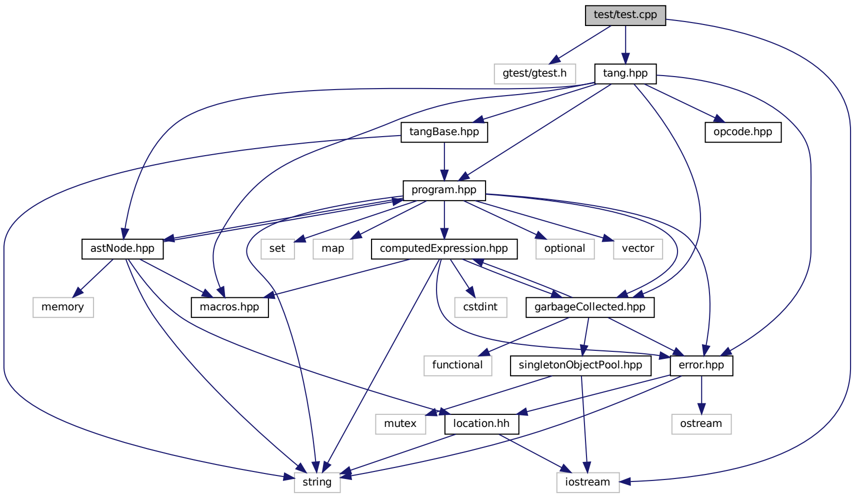
Contains the function declarations for the Tang::UnicodeString class and the interface to ICU.

6.93 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** (Expression, StringIndex)
- **TEST** (Expression, StringSlice)
- **TEST** (Expression, ArrayIndex)
- **TEST** (CodeBlock, Statements)
- **TEST** (Assign, Identifier)
- **TEST** (Assign, Index)
- **TEST** (Expression, ArraySlice)

- **TEST** (ControlFlow, IfElse)
- **TEST** (ControlFlow, While)
- **TEST** (ControlFlow, Break)
- **TEST** (ControlFlow, Continue)
- **TEST** (ControlFlow, DoWhile)
- **TEST** (ControlFlow, For)
- **TEST** (ControlFlow, RangedFor)
- **TEST** (Print, Default)
- **TEST** (Print, Array)

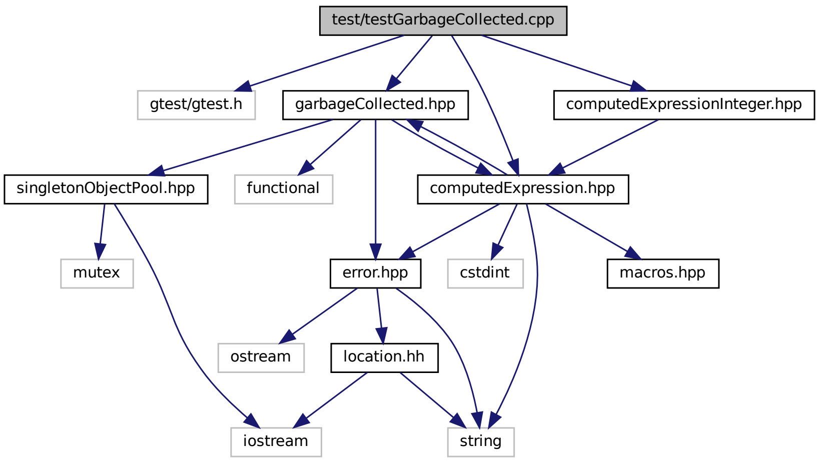
6.93.1 Detailed Description

Test the general language behaviors.

6.94 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)

6.94.1 Detailed Description

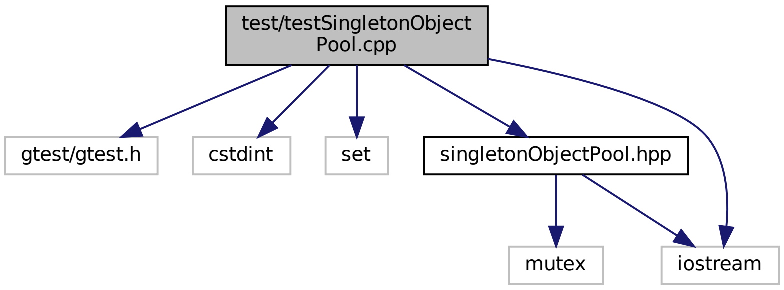
Test the generic behavior of the Tang::GarbageCollected class.

6.95 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)

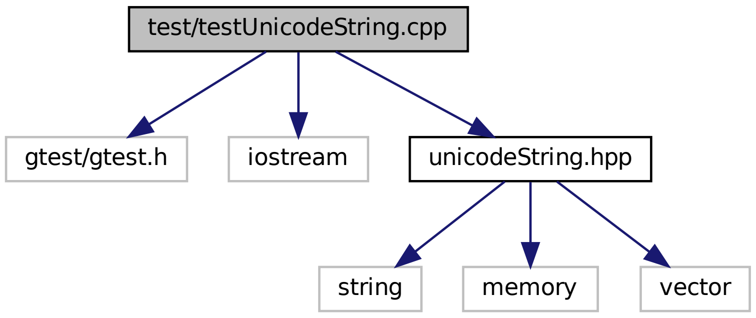
6.95.1 Detailed Description

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

6.96 test/testUnicodeString.cpp File Reference

Contains tests for the Tang::UnicodeString class.

```
#include <gtest/gtest.h>
#include <iostream>
#include "unicodeString.hpp"
```
Include dependency graph for testUnicodeString.cpp:



Functions

- **TEST** (Core, Unescape)
- **TEST** (Core, HtmlEscape)
- **TEST** (Core, HtmlEscapeAscii)
- **TEST** (UnicodeString, SubString)
- int **main** (int argc, char $**$argv)

6.96.1 Detailed Description

Contains tests for the Tang::UnicodeString class.

Index