

Język programowania

Generated by Doxygen 1.9.1

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 \$NameState Class Reference	7
4.1.1 Constructor & Destructor Documentation	8
4.1.1.1 \$NameState()	8
4.1.2 Member Function Documentation	8
4.1.2.1 parse()	8
4.2 \$ValueState Class Reference	9
4.2.1 Constructor & Destructor Documentation	9
4.2.1.1 \$ValueState()	10
4.2.2 Member Function Documentation	10
4.2.2.1 parse()	10
4.3 Base Class Reference	10
4.3.1 Constructor & Destructor Documentation	11
4.3.1.1 Base()	11
4.3.2 Member Function Documentation	11
4.3.2.1 parse()	11
4.4 Conditional Class Reference	12
4.4.1 Constructor & Destructor Documentation	13
4.4.1.1 Conditional()	13
4.4.2 Member Function Documentation	13
4.4.2.1 parse()	13
4.5 Error Class Reference	14
4.5.1 Constructor & Destructor Documentation	14
4.5.1.1 Error()	15
4.5.2 Member Function Documentation	15
4.5.2.1 parse()	15
4.6 FirstRowState Class Reference	15
4.6.1 Detailed Description	16
4.6.2 Constructor & Destructor Documentation	16
4.6.2.1 FirstRowState()	16
4.6.3 Member Function Documentation	16
4.6.3.1 parse()	17
4.7 Function Class Reference	17
4.7.1 Constructor & Destructor Documentation	18

4.7.1.1 Function()	18
4.7.2 Member Function Documentation	18
4.7.2.1 get_name()	18
4.7.2.2 get_type()	19
4.7.2.3 get_value()	19
4.8 FunctionCall Class Reference	19
4.8.1 Detailed Description	20
4.8.2 Constructor & Destructor Documentation	20
4.8.2.1 FunctionCall()	20
4.8.2.2 ~FunctionCall()	20
4.8.3 Member Function Documentation	20
4.8.3.1 parse()	21
4.9 Matrix Class Reference	21
4.9.1 Constructor & Destructor Documentation	22
4.9.1.1 Matrix() [1/2]	22
4.9.1.2 Matrix() [2/2]	22
4.9.2 Member Function Documentation	22
4.9.2.1 add_column()	22
4.9.2.2 add_row()	23
4.9.2.3 add_value()	23
4.9.2.4 get()	23
4.9.2.5 get_from_stack()	24
4.9.2.6 is_matrix()	24
4.9.2.7 operator() [1/2]	25
4.9.2.8 operator() [2/2]	25
4.9.2.9 operator==()	25
4.9.2.10 parse_matrix()	25
4.9.2.11 repr()	26
4.9.2.12 size()	27
4.9.2.13 translate()	27
4.10 Parser Class Reference	27
4.10.1 Constructor & Destructor Documentation	28
4.10.1.1 Parser()	28
4.10.2 Member Function Documentation	28
4.10.2.1 parse_string()	28
4.10.3 Member Data Documentation	28
4.10.3.1 stack_	29
4.11 RowState Class Reference	29
4.11.1 Detailed Description	30
4.11.2 Constructor & Destructor Documentation	30
4.11.2.1 RowState()	30
4.11.3 Member Function Documentation	30

4.11.3.1 parse()	30
4.12 Scope Class Reference	31
4.12.1 Constructor & Destructor Documentation	31
4.12.1.1 Scope()	32
4.12.2 Member Function Documentation	32
4.12.2.1 get_name()	32
4.12.2.2 get_type()	32
4.13 State Class Reference	33
4.13.1 Constructor & Destructor Documentation	33
4.13.1.1 State()	34
4.13.2 Member Function Documentation	34
4.13.2.1 parse()	34
4.13.3 Member Data Documentation	34
4.13.3.1 stack_	34
4.14 Token Class Reference	35
4.14.1 Detailed Description	35
4.14.2 Member Enumeration Documentation	35
4.14.2.1 TokenType	35
4.14.3 Member Function Documentation	36
4.14.3.1 get_name()	36
4.14.3.2 get_type()	36
4.15 Utility Class Reference	36
4.15.1 Member Function Documentation	37
4.15.1.1 find_token()	37
4.15.1.2 whitespace()	37
4.16 Variable Class Reference	38
4.16.1 Constructor & Destructor Documentation	39
4.16.1.1 Variable()	39
4.16.2 Member Function Documentation	39
4.16.2.1 get_name()	39
4.16.2.2 get_type()	40
4.16.2.3 get_value()	40
4.16.2.4 set_value()	40
4.17 VariableAssignment Class Reference	41
4.17.1 Detailed Description	42
4.17.2 Constructor & Destructor Documentation	42
4.17.2.1 VariableAssignment()	42
4.17.3 Member Function Documentation	42
4.17.3.1 parse()	42
5 File Documentation	45
5.1 main.cpp File Reference	45

5.1.1 Function Documentation	46
5.1.1.1 eq()	46
5.1.1.2 exit_func()	46
5.1.1.3 hello()	47
5.1.1.4 input()	47
5.1.1.5 main()	48
5.1.1.6 newline()	48
5.1.1.7 not_func()	49
5.1.1.8 ones()	49
5.1.1.9 print()	50
5.1.1.10 text()	50
5.2 Matrix.cpp File Reference	51
5.3 Matrix.h File Reference	52
5.4 state-machine/Parser.h File Reference	53
5.5 state-machine/State.h File Reference	54
5.5.1 Macro Definition Documentation	56
5.5.1.1 CHANGE_STATE	56
5.6 state-machine/states/Base.h File Reference	56
5.7 state-machine/states/Conditional.h File Reference	57
5.8 state-machine/states/CreateVariable.h File Reference	58
5.9 state-machine/states/Error.h File Reference	60
5.10 state-machine/states/FunctionCall.h File Reference	60
5.11 state-machine/states/VariableAssignment.h File Reference	61
5.12 tokens/Function.h File Reference	62
5.12.1 Macro Definition Documentation	64
5.12.1.1 FUNCTION	64
5.13 tokens/Scope.h File Reference	64
5.14 tokens/Token.h File Reference	66
5.14.1 Typedef Documentation	66
5.14.1.1 Stack	66
5.15 tokens/Variable.h File Reference	67
5.15.1 Typedef Documentation	68
5.15.1.1 ValueType	68
5.16 Utility.h File Reference	68
Index	71

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Matrix	21
Parser	27
State	33
\$NameState	7
\$ValueState	9
Base	10
Conditional	12
Error	14
FirstRowState	15
FunctionCall	19
RowState	29
VariableAssignment	41
Token	35
Function	17
Scope	31
Variable	38
Utility	36

Chapter 2

Class Index

2.1 Class List

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

\$NameState	7
\$ValueState	9
Base	10
Conditional	12
Error	14
FirstRowState	
State used for creating first row in Matrix	15
Function	17
FunctionCall	
State used for function calls	19
Matrix	21
Parser	27
RowState	
State used for creating rows in matrix	29
Scope	31
State	33
Token	
Interface for all the Tokens encounter in the code	35
Utility	36
Variable	38
VariableAssignment	
State used for parsing variables	41

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

main.cpp	45
Matrix.cpp	51
Matrix.h	52
Utility.h	68
state-machine/ Parser.h	53
state-machine/ State.h	54
state-machine/states/ Base.h	56
state-machine/states/ Conditional.h	57
state-machine/states/ CreateVariable.h	58
state-machine/states/ Error.h	60
state-machine/states/ FunctionCall.h	60
state-machine/states/ VariableAssignment.h	61
tokens/ Function.h	62
tokens/ Scope.h	64
tokens/ Token.h	66
tokens/ Variable.h	67

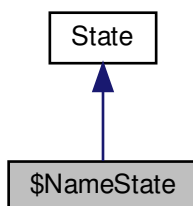
Chapter 4

Class Documentation

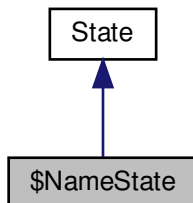
4.1 \$NameState Class Reference

```
#include <CreateVariable.h>
```

Inheritance diagram for \$NameState:



Collaboration diagram for \$NameState:



Public Member Functions

- CreateVariable [\\$NameState](#) ([Stack](#) &stack)
- [State](#) * [parse](#) (const std::string &text, int position) override

Additional Inherited Members

4.1.1 Constructor & Destructor Documentation

4.1.1.1 [\\$NameState](#)()

```
CreateVariable $NameState::$NameState (
    Stack & stack ) [inline]
```

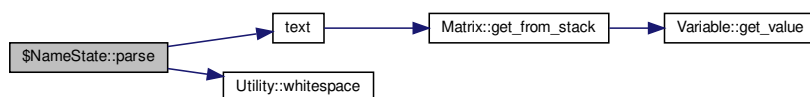
4.1.2 Member Function Documentation

4.1.2.1 [parse](#)()

```
State* $NameState::parse (
    const std::string & text,
    int position ) [inline], [override], [virtual]
```

Implements [State](#).

Here is the call graph for this function:



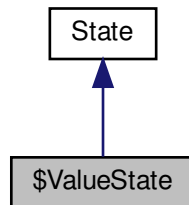
The documentation for this class was generated from the following file:

- state-machine/states/[CreateVariable.h](#)

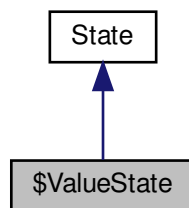
4.2 \$ValueState Class Reference

```
#include <CreateVariable.h>
```

Inheritance diagram for \$ValueState:



Collaboration diagram for \$ValueState:



Public Member Functions

- CreateVariable [\\$ValueState](#) (const std::string &name, [Stack](#) &stack)
- [State](#) * [parse](#) (const std::string &text, int position) override

Additional Inherited Members

4.2.1 Constructor & Destructor Documentation

4.2.1.1 \$ValueState()

```
CreateVariable $ValueState::$ValueState (
    const std::string & name,
    Stack & stack ) [inline]
```

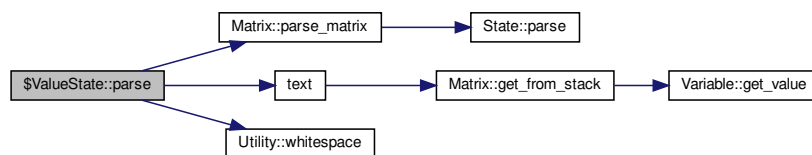
4.2.2 Member Function Documentation

4.2.2.1 parse()

```
State* $ValueState::parse (
    const std::string & text,
    int position ) [inline], [override], [virtual]
```

Implements [State](#).

Here is the call graph for this function:



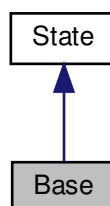
The documentation for this class was generated from the following file:

- state-machine/states/[CreateVariable.h](#)

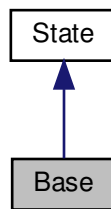
4.3 Base Class Reference

```
#include <Base.h>
```

Inheritance diagram for Base:



Collaboration diagram for Base:



Public Member Functions

- [Base](#) ([Stack](#) &stack)
- [State](#) * [parse](#) (const std::string &text, int position) override

Additional Inherited Members

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Base()

```
Base::Base (  
    Stack & stack ) [inline]
```

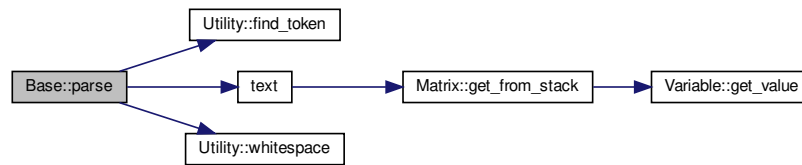
4.3.2 Member Function Documentation

4.3.2.1 parse()

```
State* Base::parse (  
    const std::string & text,  
    int position ) [inline], [override], [virtual]
```

Implements [State](#).

Here is the call graph for this function:



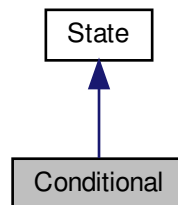
The documentation for this class was generated from the following file:

- state-machine/states/[Base.h](#)

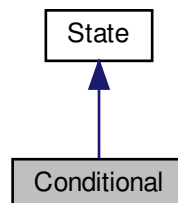
4.4 Conditional Class Reference

```
#include <Conditional.h>
```

Inheritance diagram for Conditional:



Collaboration diagram for Conditional:



Public Member Functions

- [Conditional](#) ([Stack](#) &stack)
- [State](#) * [parse](#) (const std::string &text, int position) override

Additional Inherited Members

4.4.1 Constructor & Destructor Documentation

4.4.1.1 Conditional()

```
Conditional::Conditional (
    Stack & stack ) [inline]
```

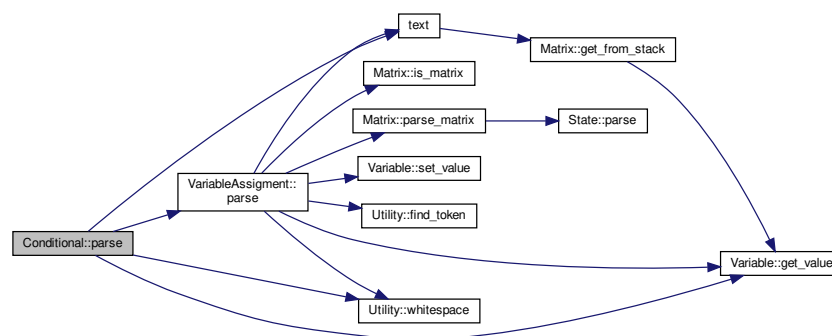
4.4.2 Member Function Documentation

4.4.2.1 parse()

```
State* Conditional::parse (
    const std::string & text,
    int position ) [inline], [override], [virtual]
```

Implements [State](#).

Here is the call graph for this function:



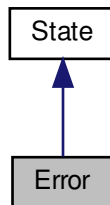
The documentation for this class was generated from the following file:

- state-machine/states/[Conditional.h](#)

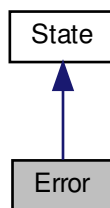
4.5 Error Class Reference

```
#include <Error.h>
```

Inheritance diagram for Error:



Collaboration diagram for Error:



Public Member Functions

- [Error](#) (const std::string &error, [Stack](#) &stack)
- [State](#) * [parse](#) (const std::string &[text](#), int position) override

Additional Inherited Members

4.5.1 Constructor & Destructor Documentation

4.5.1.1 Error()

```
Error::Error (
    const std::string & error,
    Stack & stack ) [inline], [explicit]
```

4.5.2 Member Function Documentation

4.5.2.1 parse()

```
State* Error::parse (
    const std::string & text,
    int position ) [inline], [override], [virtual]
```

Implements [State](#).

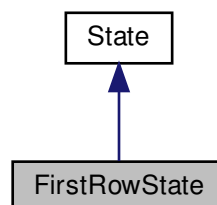
The documentation for this class was generated from the following file:

- state-machine/states/[Error.h](#)

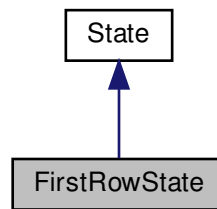
4.6 FirstRowState Class Reference

[State](#) used for creating first row in [Matrix](#).

Inheritance diagram for FirstRowState:



Collaboration diagram for FirstRowState:



Public Member Functions

- `FirstRowState` (`Stack` &`stack`, `Matrix` &`matrix`)
- `State` * `parse` (const std::string &`text`, int position) override

Additional Inherited Members

4.6.1 Detailed Description

`State` used for creating first row in `Matrix`.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 FirstRowState()

```
FirstRowState::FirstRowState (  
    Stack & stack,  
    Matrix & matrix ) [inline]
```

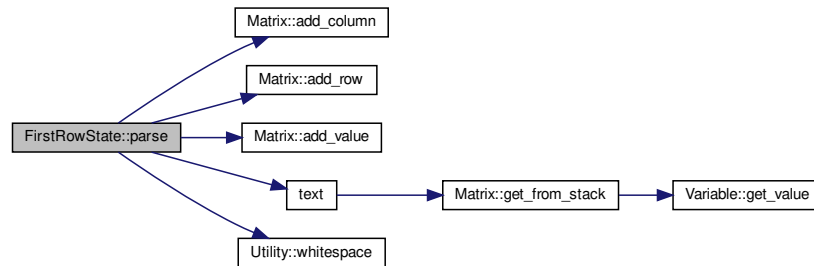
4.6.3 Member Function Documentation

4.6.3.1 parse()

```
State* FirstRowState::parse (
    const std::string & text,
    int position ) [inline], [override], [virtual]
```

Implements [State](#).

Here is the call graph for this function:



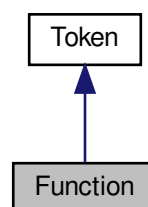
The documentation for this class was generated from the following file:

- [Matrix.cpp](#)

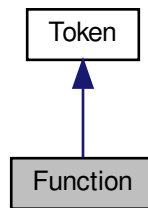
4.7 Function Class Reference

```
#include <Function.h>
```

Inheritance diagram for Function:



Collaboration diagram for Function:



Public Member Functions

- [Function](#) (std::string name, [FUNCTION](#) function)
- const std::string & [get_name](#) () override
- [TokenType](#) [get_type](#) () override
- [FUNCTION](#) [get_value](#) ()

Additional Inherited Members

4.7.1 Constructor & Destructor Documentation

4.7.1.1 Function()

```
Function::Function (
    std::string name,
    FUNCTION function ) [inline]
```

4.7.2 Member Function Documentation

4.7.2.1 get_name()

```
const std::string& Function::get_name ( ) [inline], [override], [virtual]
```

Get the name used to access the token in the code

Returns

name of the tokne in code

Implements [Token](#).

4.7.2.2 get_type()

`TokenType` `Function::get_type () [inline], [override], [virtual]`

Return the type of the token being accessed

Returns

the type of the token

Implements `Token`.

4.7.2.3 get_value()

`FUNCTION` `Function::get_value () [inline]`

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

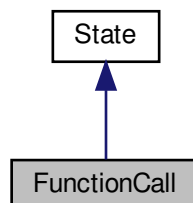
- `tokens/Function.h`

4.8 FunctionCall Class Reference

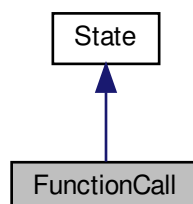
`State` used for function calls.

```
#include <FunctionCall.h>
```

Inheritance diagram for `FunctionCall`:



Collaboration diagram for `FunctionCall`:



Public Member Functions

- [FunctionCall](#) ([Stack](#) &stack, std::string buffer, [Token](#) *token)
- [State](#) * [parse](#) (const std::string &text, int position) override
- [~FunctionCall](#) ()

Additional Inherited Members

4.8.1 Detailed Description

[State](#) used for function calls.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 FunctionCall()

```
FunctionCall::FunctionCall (
    Stack & stack,
    std::string buffer,
    Token * token ) [inline]
```

4.8.2.2 ~FunctionCall()

```
FunctionCall::~~FunctionCall ( ) [inline]
```

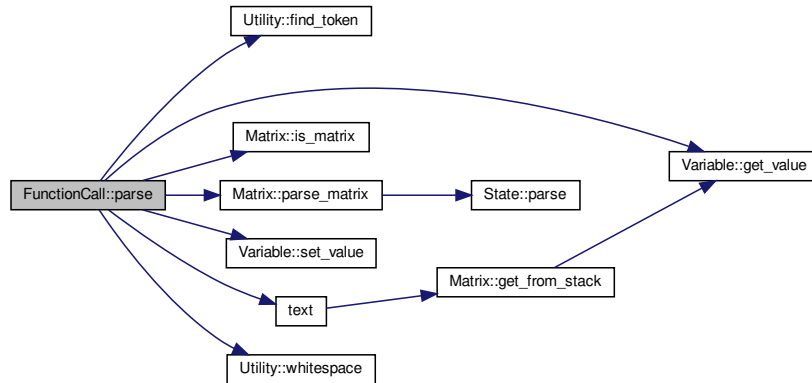
4.8.3 Member Function Documentation

4.8.3.1 parse()

```
State* FunctionCall::parse (
    const std::string & text,
    int position ) [inline], [override], [virtual]
```

Implements [State](#).

Here is the call graph for this function:



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

- state-machine/states/[FunctionCall.h](#)

4.9 Matrix Class Reference

```
#include <Matrix.h>
```

Public Member Functions

- void [add_column](#) ()
- void [add_row](#) ()
- bool [add_value](#) (double value)
- bool [operator==](#) (const [Matrix](#) &other) const
- [Matrix](#) ()
- [Matrix](#) (bool val)
- int [translate](#) (int row, int col) const
- double & [operator\(\)](#) (int idx)
- double & [operator\(\)](#) (int row, int col)
- double [get](#) (int position)
- double [size](#) ()
- std::string [repr](#) ()

Static Public Member Functions

- static bool [parse_matrix](#) (const std::string &code, [Matrix](#) &matrix)
- static bool [is_matrix](#) (const std::string buffer)
- static [Matrix](#) & [get_from_stack](#) ([Stack](#) &stack)

4.9.1 Constructor & Destructor Documentation

4.9.1.1 [Matrix\(\)](#) [1/2]

```
Matrix::Matrix ( ) [inline]
```

4.9.1.2 [Matrix\(\)](#) [2/2]

```
Matrix::Matrix (
    bool val ) [inline], [explicit]
```

4.9.2 Member Function Documentation

4.9.2.1 [add_column\(\)](#)

```
void Matrix::add_column ( ) [inline]
```

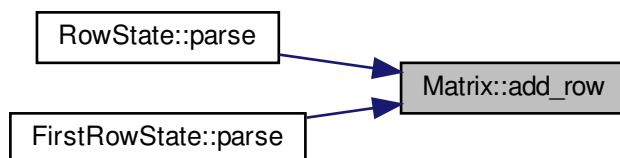
Here is the caller graph for this function:



4.9.2.2 add_row()

```
void Matrix::add_row ( ) [inline]
```

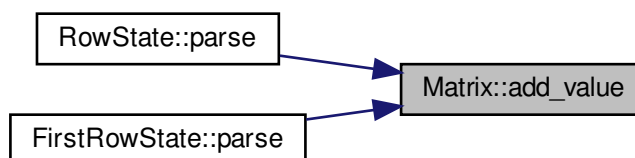
Here is the caller graph for this function:



4.9.2.3 add_value()

```
bool Matrix::add_value (
    double value ) [inline]
```

Here is the caller graph for this function:



4.9.2.4 get()

```
double Matrix::get (
    int position ) [inline]
```

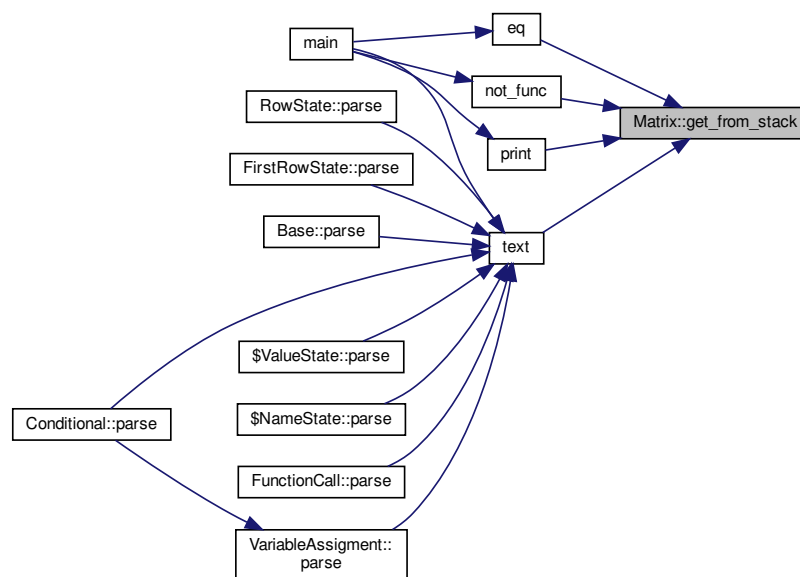
4.9.2.5 get_from_stack()

```
static Matrix& Matrix::get_from_stack (
    Stack & stack ) [inline], [static]
```

Here is the call graph for this function:



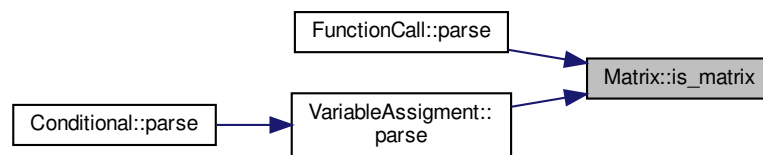
Here is the caller graph for this function:



4.9.2.6 is_matrix()

```
static bool Matrix::is_matrix (
    const std::string buffer ) [inline], [static]
```

Here is the caller graph for this function:



4.9.2.7 operator>() [1/2]

```
double& Matrix::operator() (
    int idx ) [inline]
```

4.9.2.8 operator>() [2/2]

```
double& Matrix::operator() (
    int row,
    int col ) [inline]
```

Here is the call graph for this function:



4.9.2.9 operator==()

```
bool Matrix::operator==(
    const Matrix & other ) const [inline]
```

4.9.2.10 parse_matrix()

```
bool Matrix::parse_matrix (
    const std::string & code,
    Matrix & matrix ) [static]
```

Function that parses a matrix and assigns it into @matrix

Parameters

<i>code</i>	code to be parsed
<i>matrix</i>	matrix to bullied up

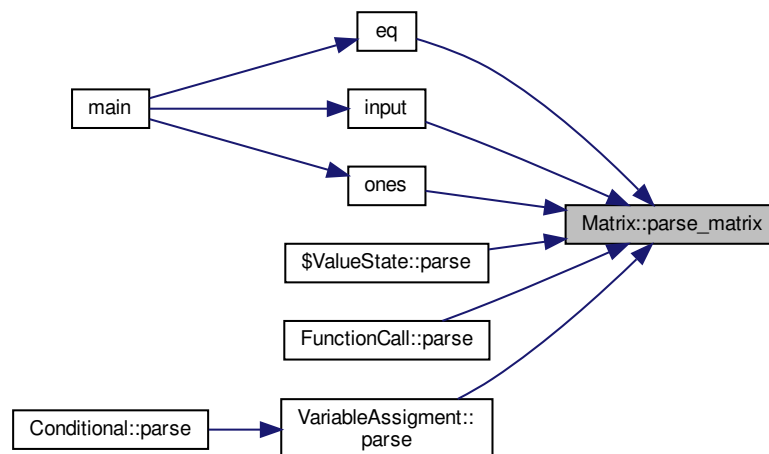
Returns

true if parsing completed successfully and false if it failed

Here is the call graph for this function:



Here is the caller graph for this function:

**4.9.2.11 repr()**

```
std::string Matrix::repr ( )
```

Create string representing the matrix

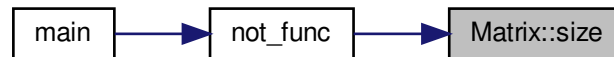
Returns

string representation of the matrix

4.9.2.12 size()

```
double Matrix::size ( ) [inline]
```

Here is the caller graph for this function:



4.9.2.13 translate()

```
int Matrix::translate (
    int row,
    int col ) const [inline]
```

Here is the caller graph for this function:



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

- [Matrix.h](#)
- [Matrix.cpp](#)

4.10 Parser Class Reference

```
#include <Parser.h>
```

Public Member Functions

- [Parser](#) ()
- void [parse_string](#) (const std::string &code)

Public Attributes

- [Stack stack_](#)

4.10.1 Constructor & Destructor Documentation

4.10.1.1 Parser()

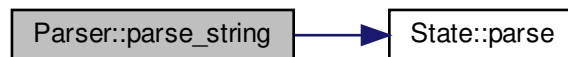
```
Parser::Parser ( ) [inline]
```

4.10.2 Member Function Documentation

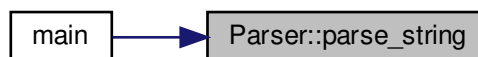
4.10.2.1 parse_string()

```
void Parser::parse_string (
    const std::string & code ) [inline]
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.10.3 Member Data Documentation

4.10.3.1 stack_

`Stack Parser::stack_`

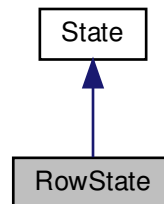
The documentation for this class was generated from the following file:

- state-machine/[Parser.h](#)

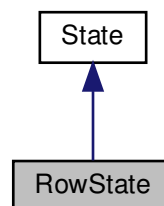
4.11 RowState Class Reference

[State](#) used for creating rows in matrix.

Inheritance diagram for RowState:



Collaboration diagram for RowState:



Public Member Functions

- [RowState](#) ([Stack](#) &stack, [Matrix](#) &matrix)
- [State](#) * [parse](#) (const std::string &text, int position) override

Additional Inherited Members

4.11.1 Detailed Description

[State](#) used for creating rows in matrix.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 RowState()

```
RowState::RowState (
    Stack & stack,
    Matrix & matrix ) [inline]
```

Here is the caller graph for this function:



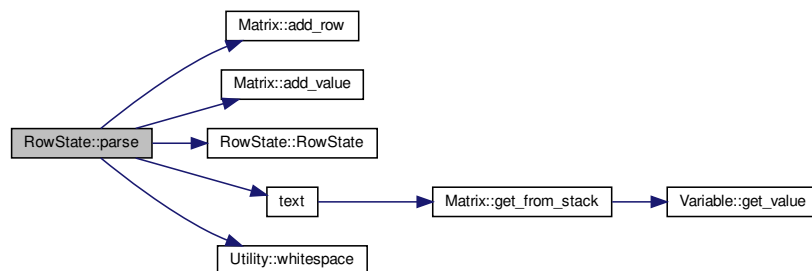
4.11.3 Member Function Documentation

4.11.3.1 parse()

```
State* RowState::parse (
    const std::string & text,
    int position ) [inline], [override], [virtual]
```

Implements [State](#).

Here is the call graph for this function:



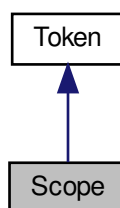
The documentation for this class was generated from the following file:

- [Matrix.cpp](#)

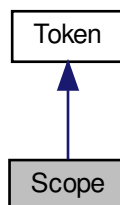
4.12 Scope Class Reference

```
#include <Scope.h>
```

Inheritance diagram for Scope:



Collaboration diagram for Scope:



Public Member Functions

- [Scope](#) (std::string name)
- const std::string & [get_name](#) () override
- [TokenType](#) [get_type](#) () override

Additional Inherited Members

4.12.1 Constructor & Destructor Documentation

4.12.1.1 Scope()

```
Scope::Scope (
    std::string name ) [inline]
```

4.12.2 Member Function Documentation

4.12.2.1 get_name()

```
const std::string& Scope::get_name ( ) [inline], [override], [virtual]
```

Get the name used to access the token in the code

Returns

name of the tokne in code

Implements [Token](#).

4.12.2.2 get_type()

```
TokenType Scope::get_type ( ) [inline], [override], [virtual]
```

Return the type of the token being accessed

Returns

the type of the token

Implements [Token](#).

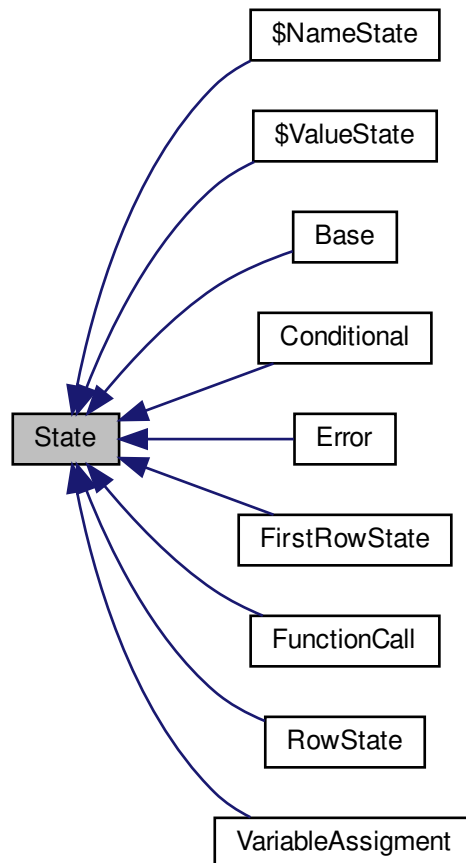
The documentation for this class was generated from the following file:

- [tokens/Scope.h](#)

4.13 State Class Reference

```
#include <State.h>
```

Inheritance diagram for State:



Public Member Functions

- `State (Stack &stack)`
- `virtual State * parse (const std::string &text, int position)=0`

Protected Attributes

- `Stack & stack_`

4.13.1 Constructor & Destructor Documentation

4.13.1.1 State()

```
State::State (
    Stack & stack ) [inline]
```

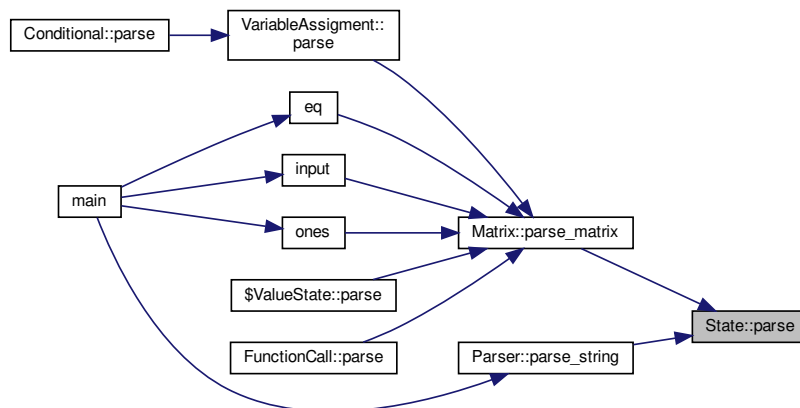
4.13.2 Member Function Documentation

4.13.2.1 parse()

```
virtual State* State::parse (
    const std::string & text,
    int position ) [pure virtual]
```

Implemented in [VariableAssignment](#), [FunctionCall](#), [Error](#), [\\$NameState](#), [\\$ValueState](#), [Conditional](#), [Base](#), [FirstRowState](#), and [RowState](#).

Here is the caller graph for this function:



4.13.3 Member Data Documentation

4.13.3.1 stack_

```
Stack& State::stack_ [protected]
```

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

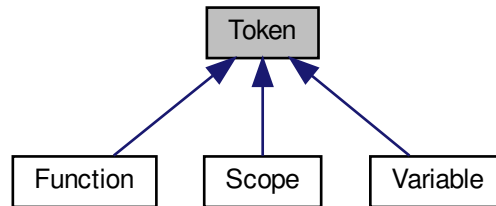
- [state-machine/State.h](#)

4.14 Token Class Reference

Interface for all the Tokens encounter in the code.

```
#include <Token.h>
```

Inheritance diagram for Token:



Public Types

- enum `TokenType` { `variable` , `function` , `scope` }
- Every token needs to know whats it type is.*

Public Member Functions

- virtual const std::string & `get_name` ()=0
- virtual `TokenType` `get_type` ()=0

4.14.1 Detailed Description

Interface for all the Tokens encounter in the code.

4.14.2 Member Enumeration Documentation

4.14.2.1 TokenType

```
enum Token::TokenType
```

Every token needs to know whats it type is.

Enumerator

variable	
function	
scope	

4.14.3 Member Function Documentation

4.14.3.1 `get_name()`

```
virtual const std::string& Token::get_name ( ) [pure virtual]
```

Get the name used to access the token in the code

Returns

name of the tokne in code

Implemented in [Variable](#), [Scope](#), and [Function](#).

4.14.3.2 `get_type()`

```
virtual TokenType Token::get_type ( ) [pure virtual]
```

Return the type of the token being accessed

Returns

the type of the token

Implemented in [Variable](#), [Scope](#), and [Function](#).

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

- [tokens/Token.h](#)

4.15 Utility Class Reference

```
#include <Utility.h>
```

Static Public Member Functions

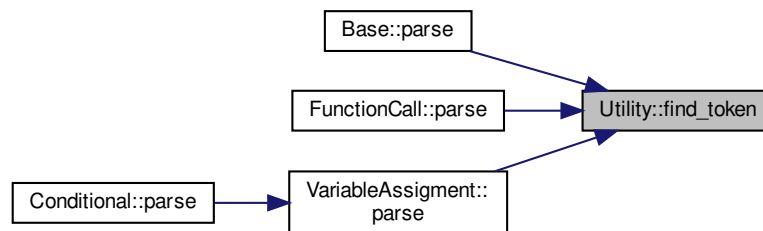
- static `Token * find_token` (const `Stack` &stack, const std::string &token_name)
- static bool `whitespace` (char letter)

4.15.1 Member Function Documentation

4.15.1.1 find_token()

```
static Token* Utility::find_token (  
    const Stack & stack,  
    const std::string & token_name ) [inline], [static]
```

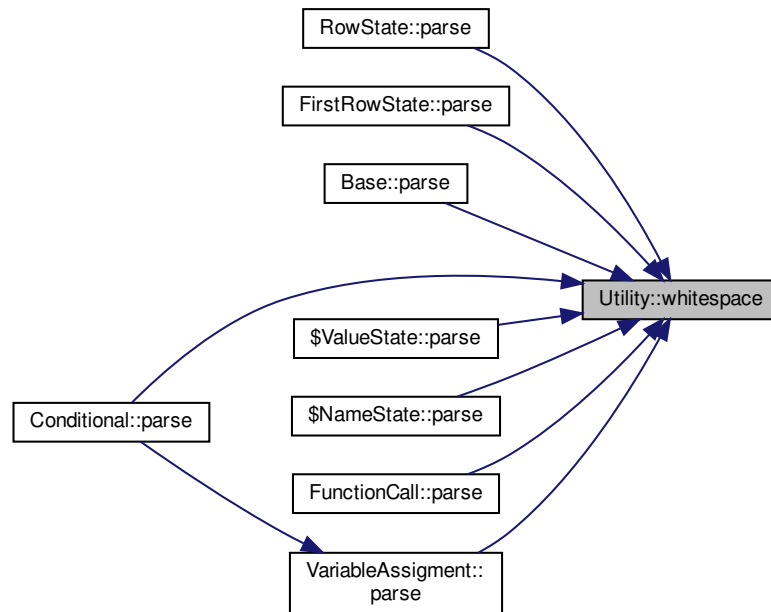
Here is the caller graph for this function:



4.15.1.2 whitespace()

```
static bool Utility::whitespace (  
    char letter ) [inline], [static]
```

Here is the caller graph for this function:



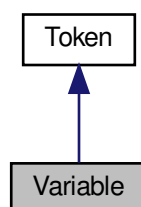
The documentation for this class was generated from the following file:

- [Utility.h](#)

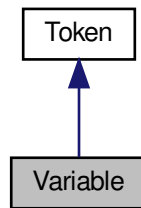
4.16 Variable Class Reference

```
#include <Variable.h>
```

Inheritance diagram for Variable:



Collaboration diagram for Variable:



Public Member Functions

- `Variable` (`std::string` name, `ValueType` value)
- `std::string & get_name` () override
- `TokenType get_type` () override
- void `set_value` (`Matrix` value)
- `Matrix & get_value` ()

Additional Inherited Members

4.16.1 Constructor & Destructor Documentation

4.16.1.1 Variable()

```
Variable::Variable (
    std::string name,
    ValueType value ) [inline]
```

4.16.2 Member Function Documentation

4.16.2.1 get_name()

```
std::string& Variable::get_name ( ) [inline], [override], [virtual]
```

Get the name used to access the token in the code

Returns

name of the tokne in code

Implements `Token`.

4.16.2.2 get_type()

```
TokenType Variable::get_type ( ) [inline], [override], [virtual]
```

Return the type of the token being accessed

Returns

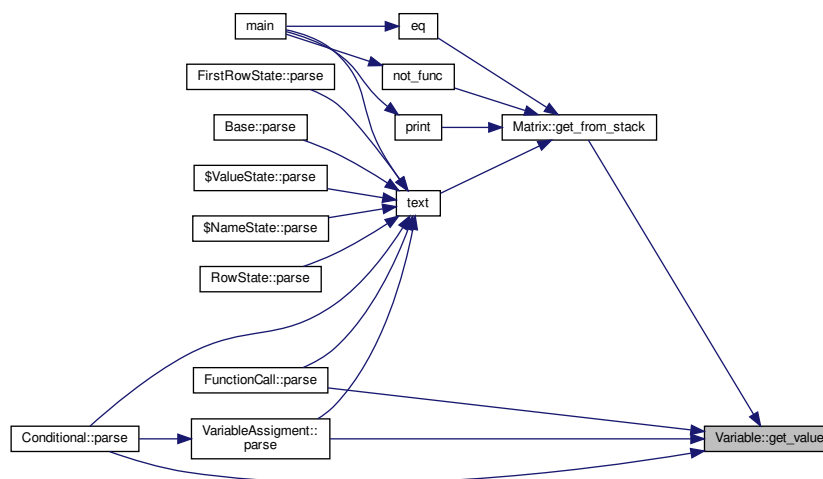
the type of the token

Implements [Token](#).

4.16.2.3 get_value()

```
Matrix& Variable::get_value ( ) [inline]
```

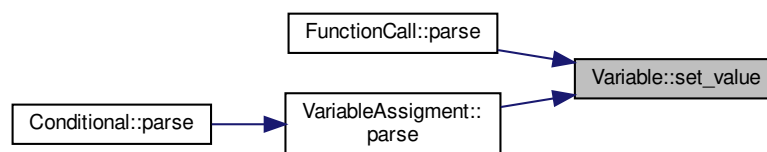
Here is the caller graph for this function:



4.16.2.4 set_value()

```
void Variable::set_value (
    Matrix value ) [inline]
```

Here is the caller graph for this function:



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

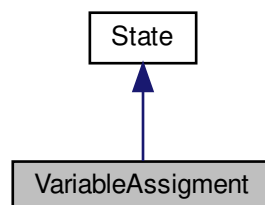
- [tokens/Variable.h](#)

4.17 VariableAssignment Class Reference

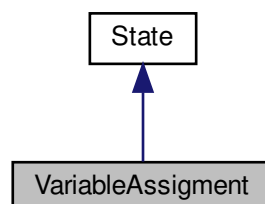
[State](#) used for parsing variables.

```
#include <VariableAssignment.h>
```

Inheritance diagram for VariableAssignment:



Collaboration diagram for VariableAssignment:



Public Member Functions

- [VariableAssignment](#) ([Stack](#) &stack, [Variable](#) *variable)
- [State](#) * [parse](#) (const std::string &text, int position) override

Additional Inherited Members

4.17.1 Detailed Description

[State](#) used for parsing variables.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 VariableAssignment()

```
VariableAssignment::VariableAssignment (
    Stack & stack,
    Variable * variable ) [inline]
```

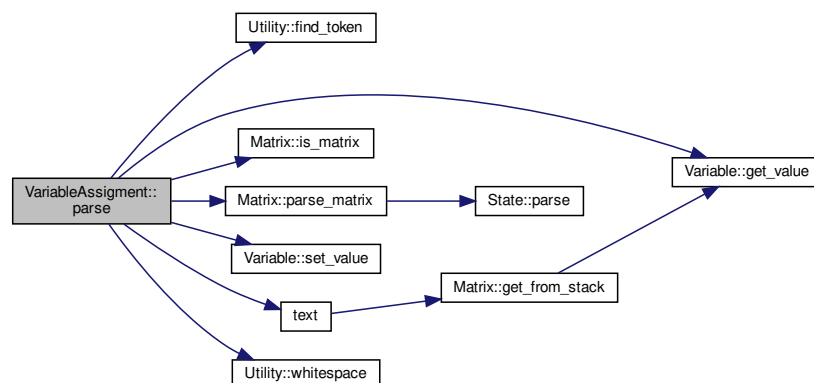
4.17.3 Member Function Documentation

4.17.3.1 parse()

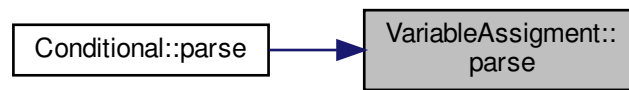
```
State* VariableAssignment::parse (
    const std::string & text,
    int position ) [inline], [override], [virtual]
```

Implements [State](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

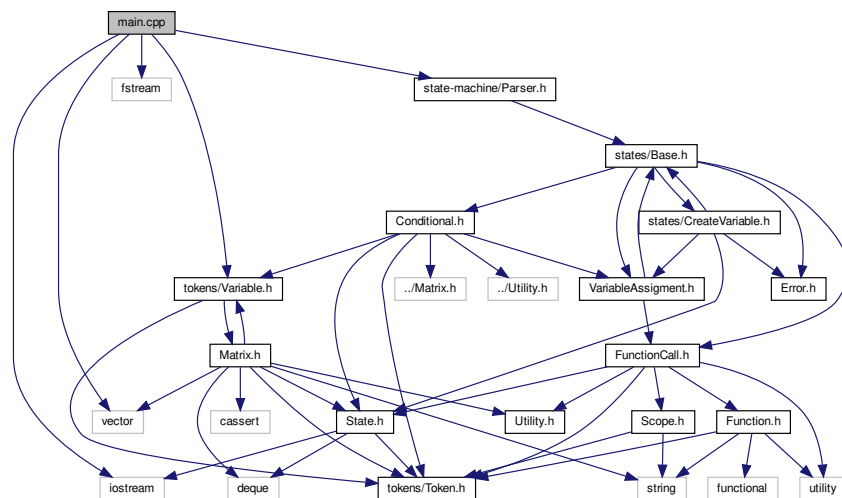
- state-machine/states/[VariableAssignment.h](#)

Chapter 5

File Documentation

5.1 main.cpp File Reference

```
#include <iostream>
#include <vector>
#include <fstream>
#include "tokens/Variable.h"
#include "state-machine/Parser.h"
Include dependency graph for main.cpp:
```



Functions

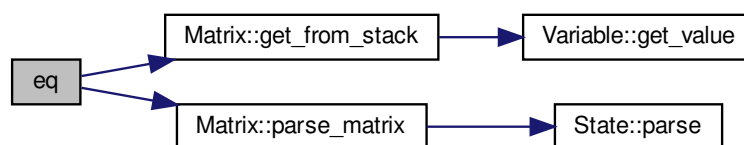
- int [hello](#) (Stack &stack)
- int [exit_func](#) (Stack &stack)
- int [print](#) (Stack &stack)
- int [ones](#) (Stack &stack)
- int [input](#) (Stack &stack)
- int [text](#) (Stack &stack)
- int [eq](#) (Stack &stack)
- int [newline](#) (Stack &stack)
- int [not_func](#) (Stack &stack)
- int [main](#) (int argc, char **argv)

5.1.1 Function Documentation

5.1.1.1 eq()

```
int eq (  
    Stack & stack )
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.1.2 exit_func()

```
int exit_func (  
    Stack & stack )
```

Here is the caller graph for this function:



5.1.1.3 hello()

```
int hello (  
    Stack & stack )
```

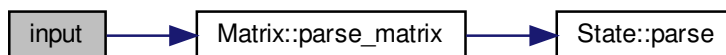
Here is the caller graph for this function:



5.1.1.4 input()

```
int input (  
    Stack & stack )
```

Here is the call graph for this function:



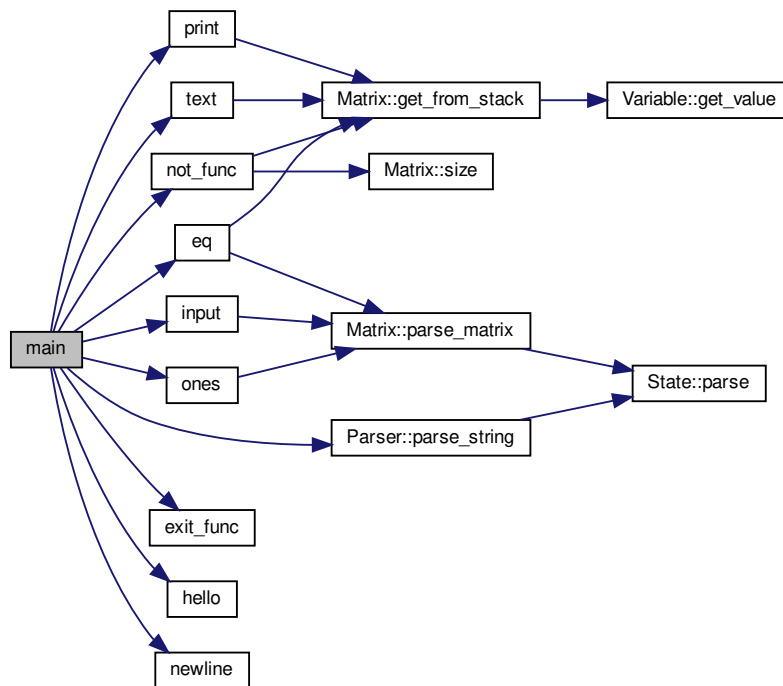
Here is the caller graph for this function:



5.1.1.5 main()

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

Here is the call graph for this function:



5.1.1.6 newline()

```
int newline (
    Stack & stack )
```

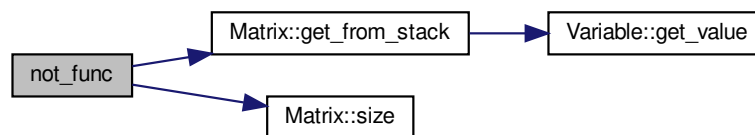
Here is the caller graph for this function:



5.1.1.7 not_func()

```
int not_func (  
    Stack & stack )
```

Here is the call graph for this function:



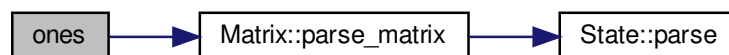
Here is the caller graph for this function:



5.1.1.8 ones()

```
int ones (  
    Stack & stack )
```

Here is the call graph for this function:



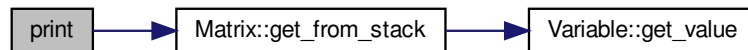
Here is the caller graph for this function:



5.1.1.9 print()

```
int print (  
    Stack & stack )
```

Here is the call graph for this function:



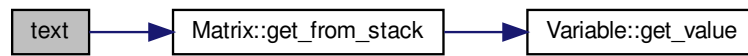
Here is the caller graph for this function:



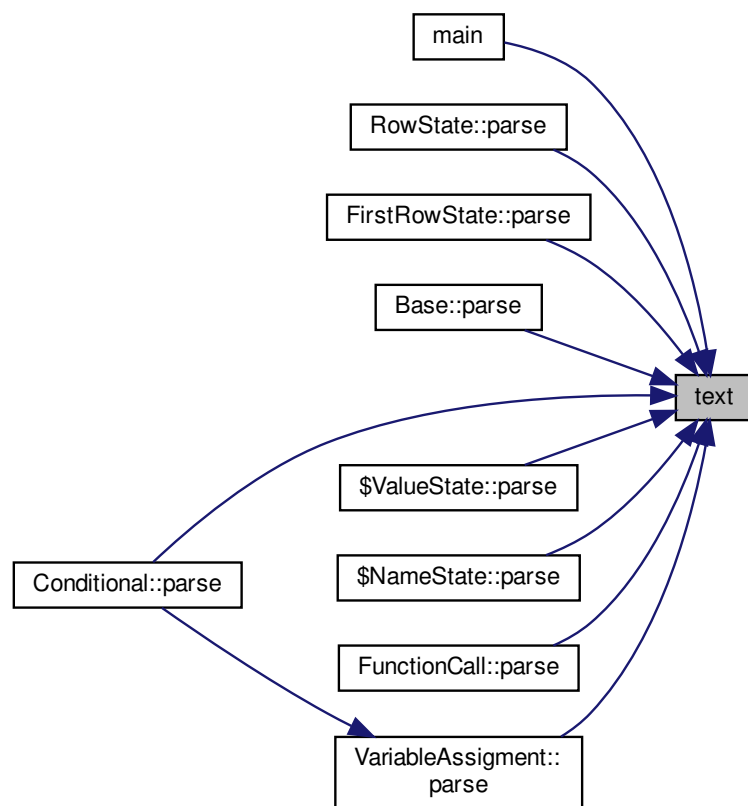
5.1.1.10 text()

```
int text (  
    Stack & stack )
```


Here is the call graph for this function:



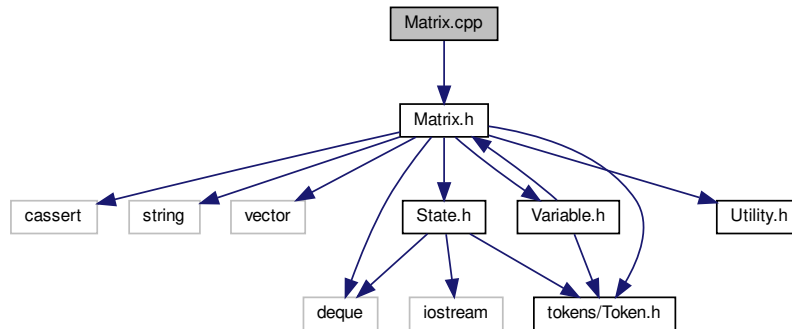
Here is the caller graph for this function:



5.2 Matrix.cpp File Reference

```
#include "Matrix.h"
```

Include dependency graph for Matrix.cpp:



Classes

- class [RowState](#)
State used for creating rows in matrix.
- class [FirstRowState](#)
State used for creating first row in [Matrix](#).

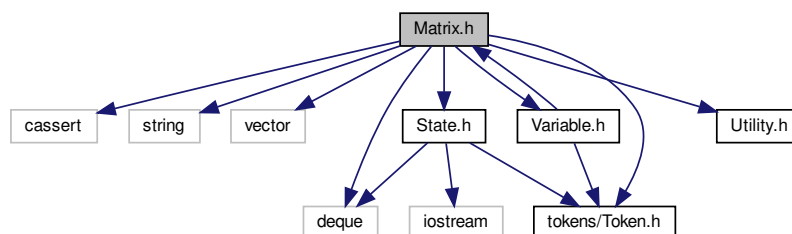
5.3 Matrix.h File Reference

```

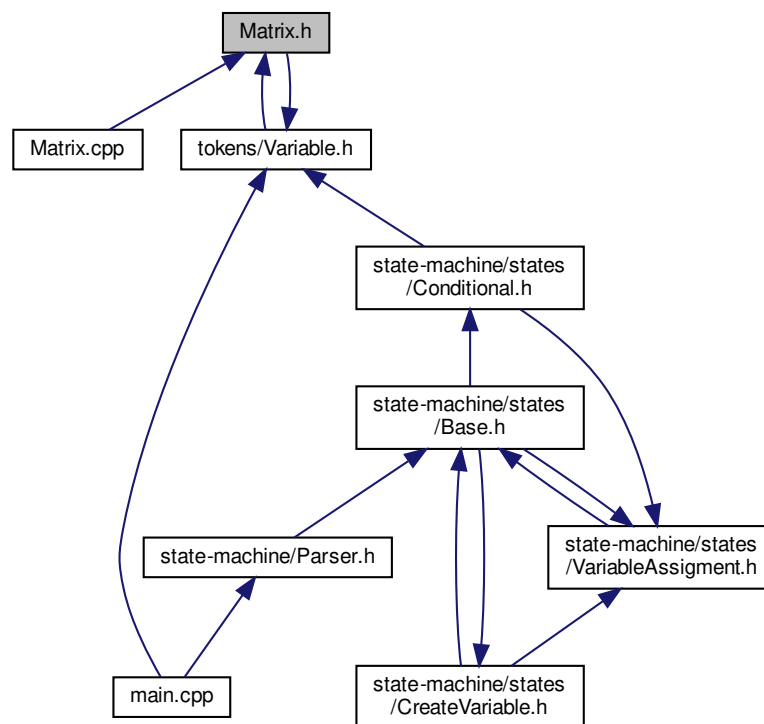
#include <cassert>
#include <string>
#include <vector>
#include <deque>
#include <Variable.h>
#include "Token.h"
#include "State.h"
#include "Utility.h"

```

Include dependency graph for Matrix.h:



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



Classes

- class [Matrix](#)

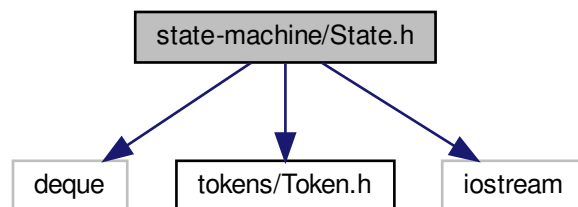
5.4 state-machine/Parser.h File Reference

```
#include "states/Base.h"
```

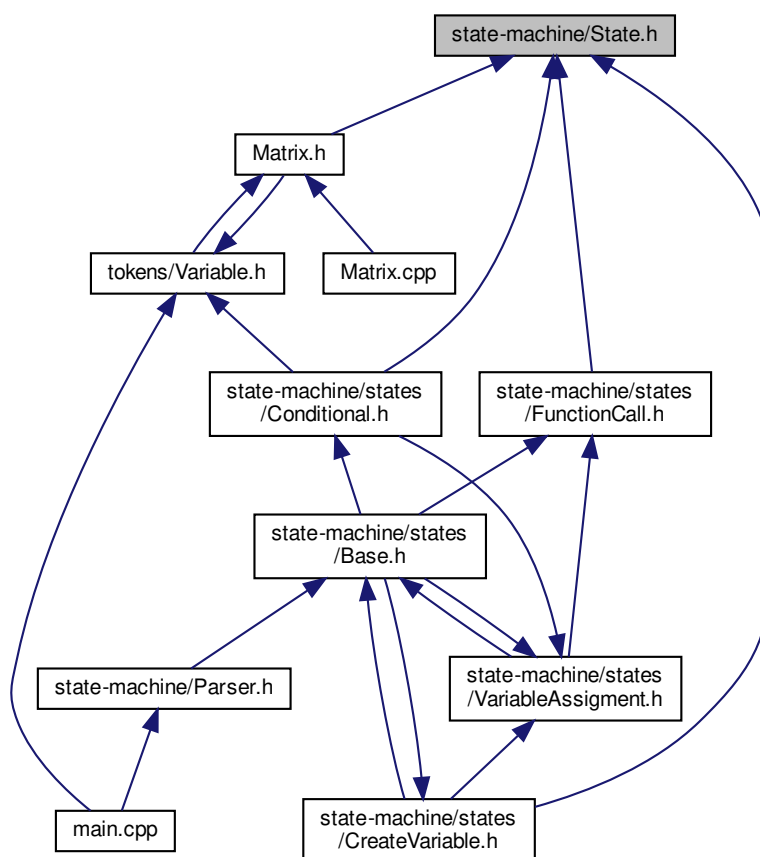


```
#include <iostream>
```

Include dependency graph for State.h:



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



Classes

- class [State](#)

Macros

- `#define CHANGE_STATE(state) std::clog << state << std::endl`

5.5.1 Macro Definition Documentation

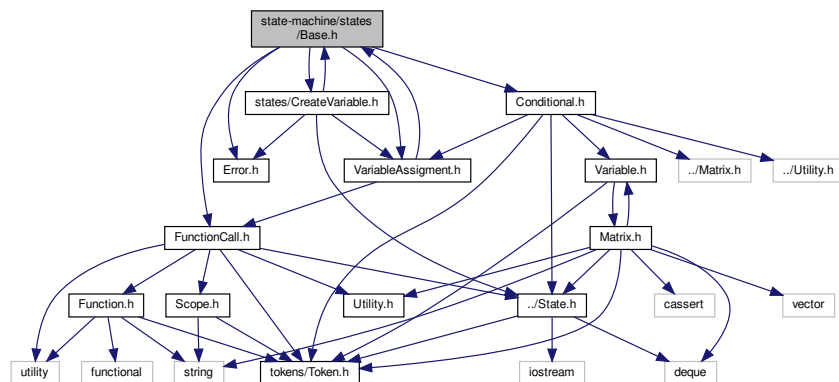
5.5.1.1 CHANGE_STATE

```
#define CHANGE_STATE(  
    state ) std::clog << state << std::endl
```

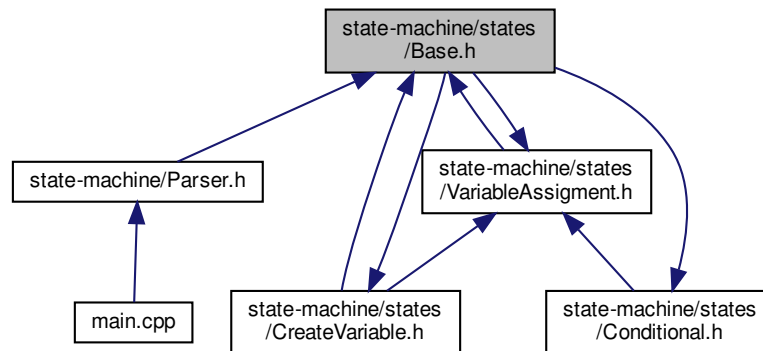
5.6 state-machine/states/Base.h File Reference

```
#include "states/CreateVariable.h"  
#include "states/VariableAssignment.h"  
#include "states/FunctionCall.h"  
#include "states/Error.h"  
#include "Conditional.h"
```

Include dependency graph for Base.h:



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



Classes

- class [Base](#)

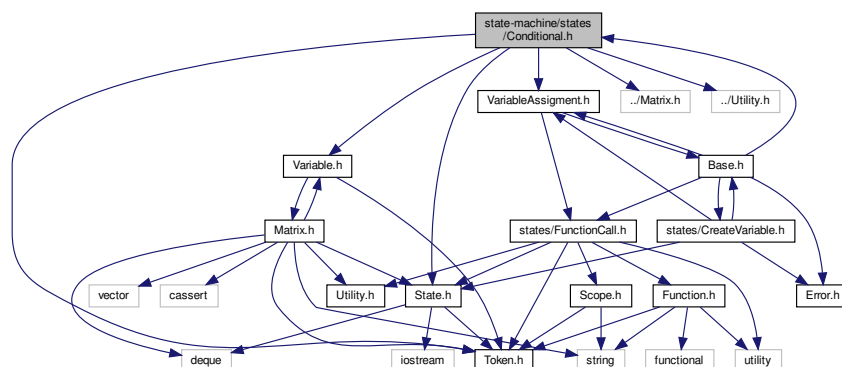
5.7 state-machine/states/Conditional.h File Reference

```

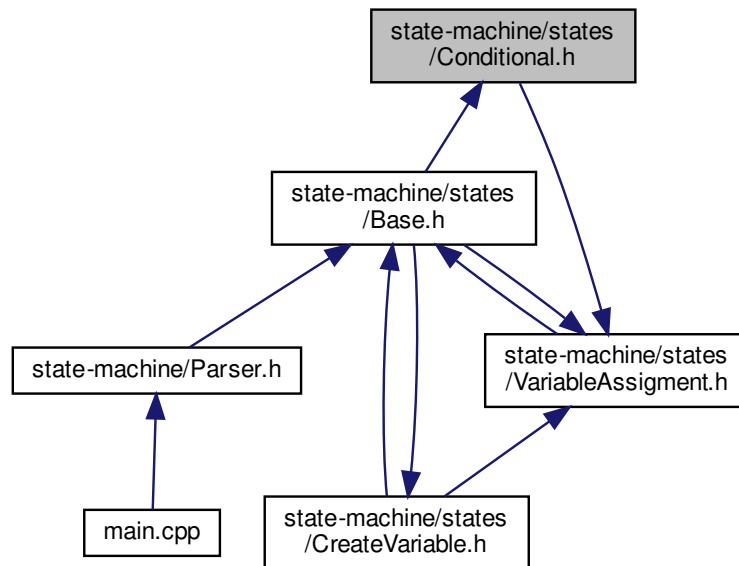
#include <Token.h>
#include "State.h"
#include "Variable.h"
#include "VariableAssignment.h"
#include "../Matrix.h"
#include "../Utility.h"

```

Include dependency graph for `Conditional.h`:



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



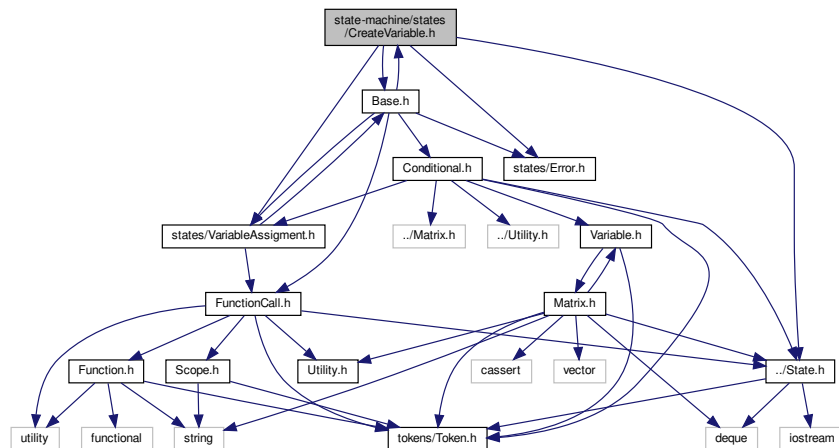
Classes

- class [Conditional](#)

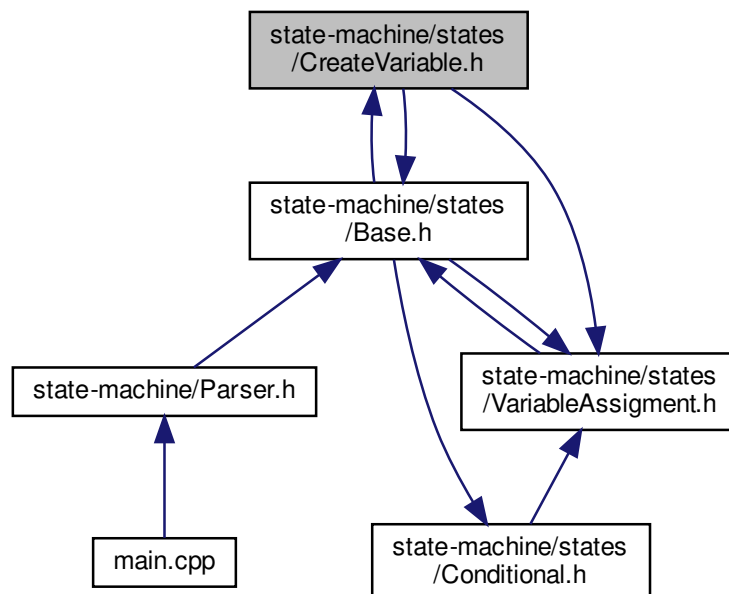
5.8 state-machine/states/CreateVariable.h File Reference

```
#include "../State.h"  
#include "Base.h"  
#include "Error.h"  
#include "VariableAssignment.h"
```


Include dependency graph for CreateVariable.h:



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

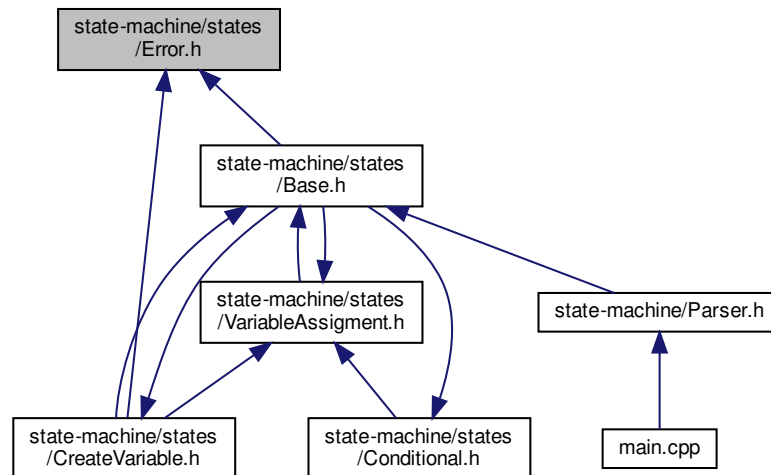


Classes

- class [\\$ValueState](#)
- class [\\$NameState](#)

5.9 state-machine/states/Error.h File Reference

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



Classes

- class [Error](#)

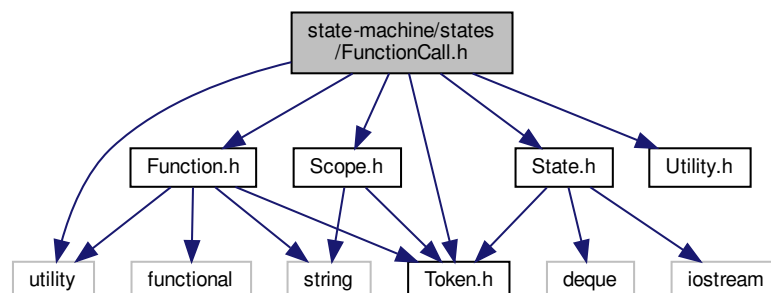
5.10 state-machine/states/FunctionCall.h File Reference

```

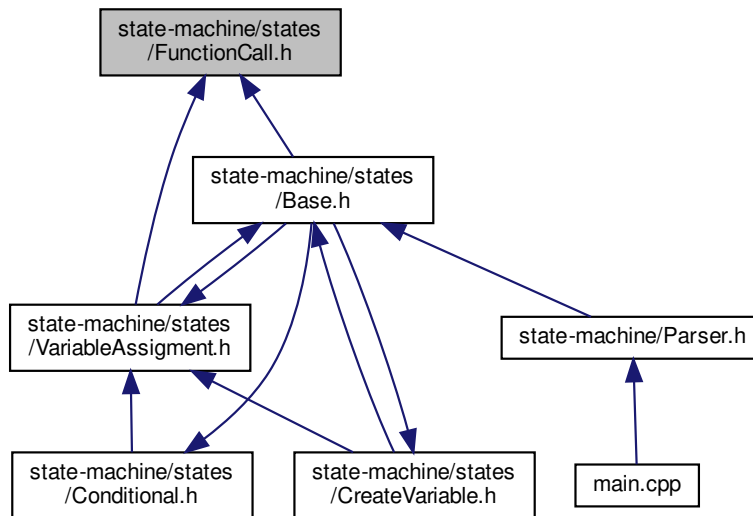
#include <utility>
#include <Function.h>
#include "Token.h"
#include "State.h"
#include "Scope.h"
#include "Utility.h"

```

Include dependency graph for `FunctionCall.h`:



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



Classes

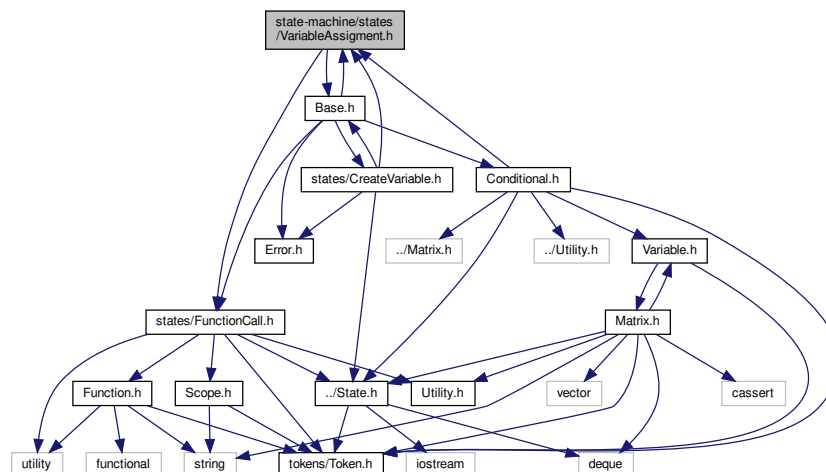
- class [FunctionCall](#)
State used for function calls.

5.11 state-machine/states/VariableAssignment.h File Reference

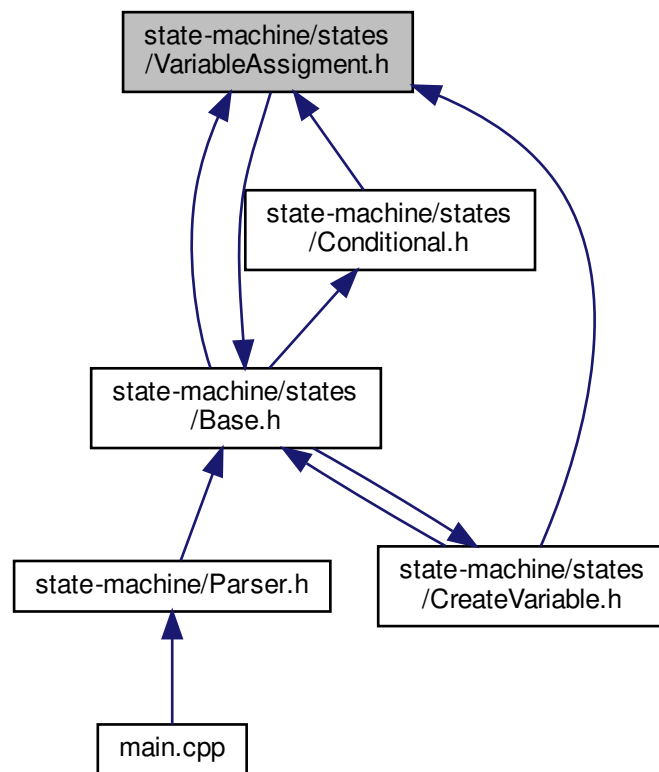
```
#include "Base.h"
```

```
#include "FunctionCall.h"
```

Include dependency graph for VariableAssignment.h:



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



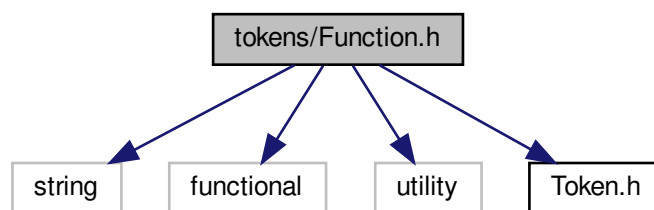
Classes

- class [VariableAssignment](#)
State used for parsing variables.

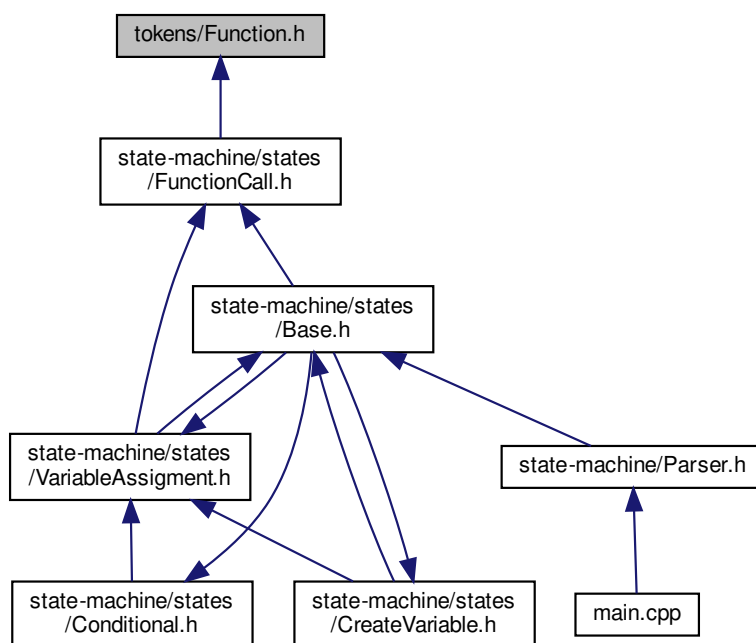
5.12 tokens/Function.h File Reference

```
#include <string>
#include <functional>
#include <utility>
#include "Token.h"
```

Include dependency graph for Function.h:



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



Classes

- class [Function](#)

Macros

- `#define FUNCTION std::function<int(Stack&)>`

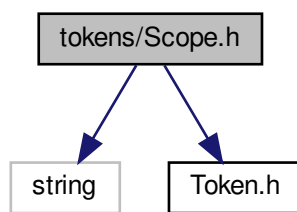
5.12.1 Macro Definition Documentation

5.12.1.1 FUNCTION

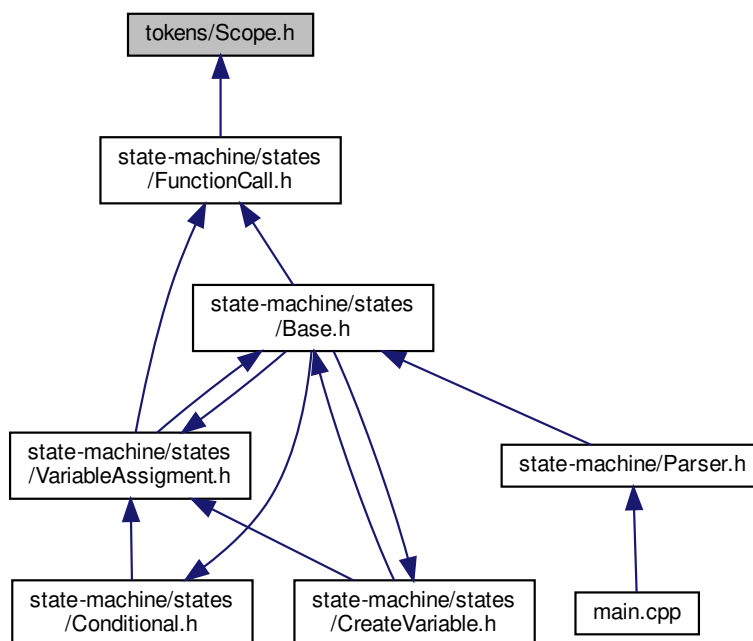
```
#define FUNCTION std::function<int (Stack&)>
```

5.13 tokens/Scope.h File Reference

```
#include <string>
#include "Token.h"
Include dependency graph for Scope.h:
```



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

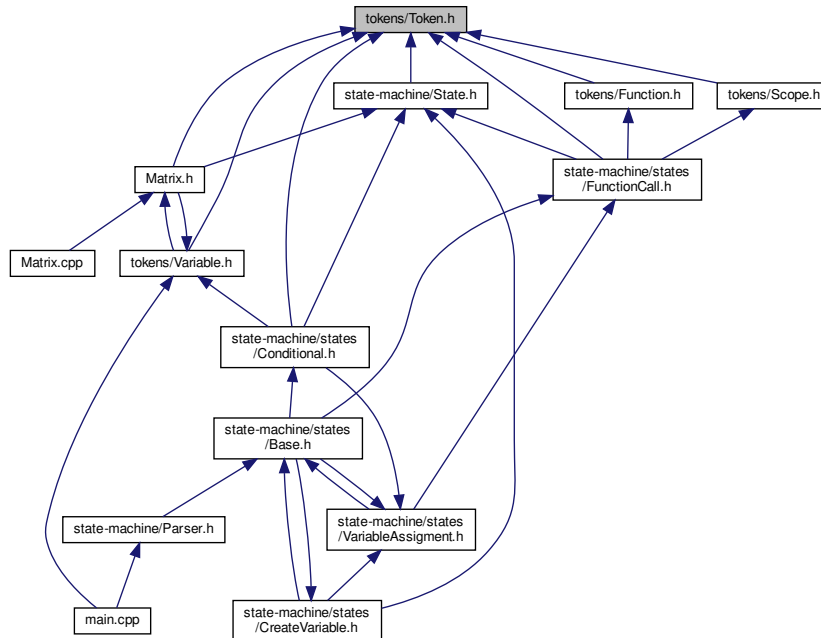


Classes

- class [Scope](#)

5.14 tokens/Token.h File Reference

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



Classes

- class [Token](#)
Interface for all the Tokens encounter in the code.

Typedefs

- using [Stack](#) = std::vector< [Token](#) * >

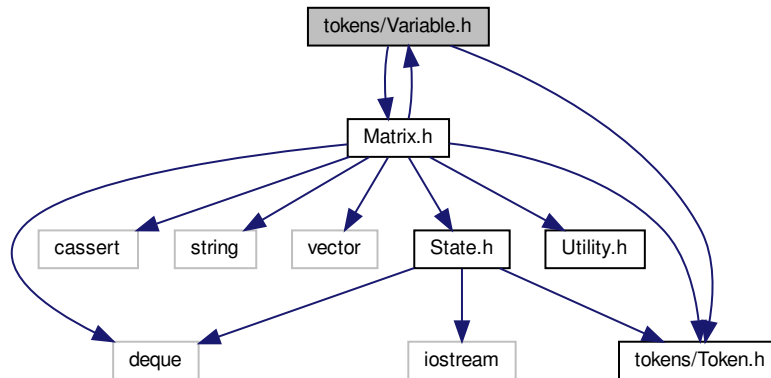
5.14.1 Typedef Documentation

5.14.1.1 Stack

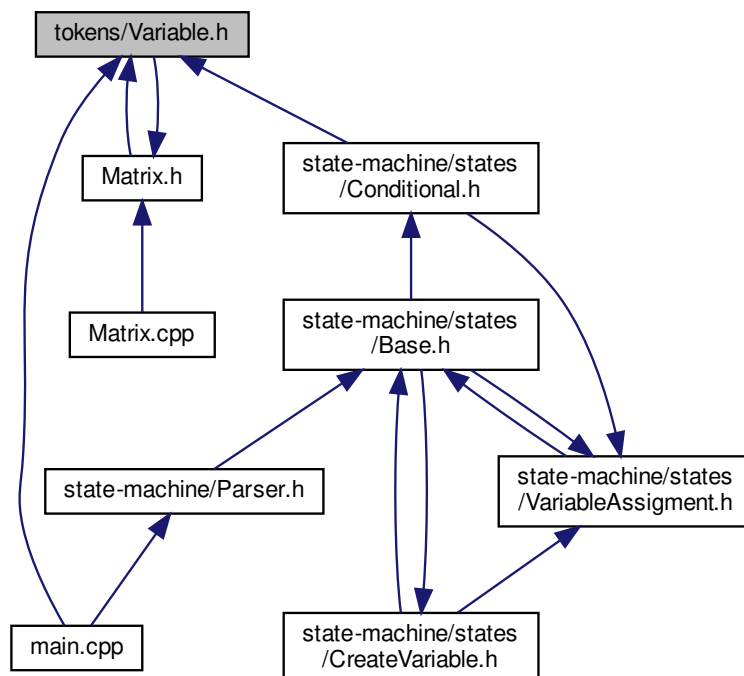
```
using Stack = std::vector<Token *>
```


5.15 tokens/Variable.h File Reference

```
#include "tokens/Token.h"
#include "Matrix.h"
Include dependency graph for Variable.h:
```



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



Classes

- class [Variable](#)

Typedefs

- using [ValueType](#) = [Matrix](#)

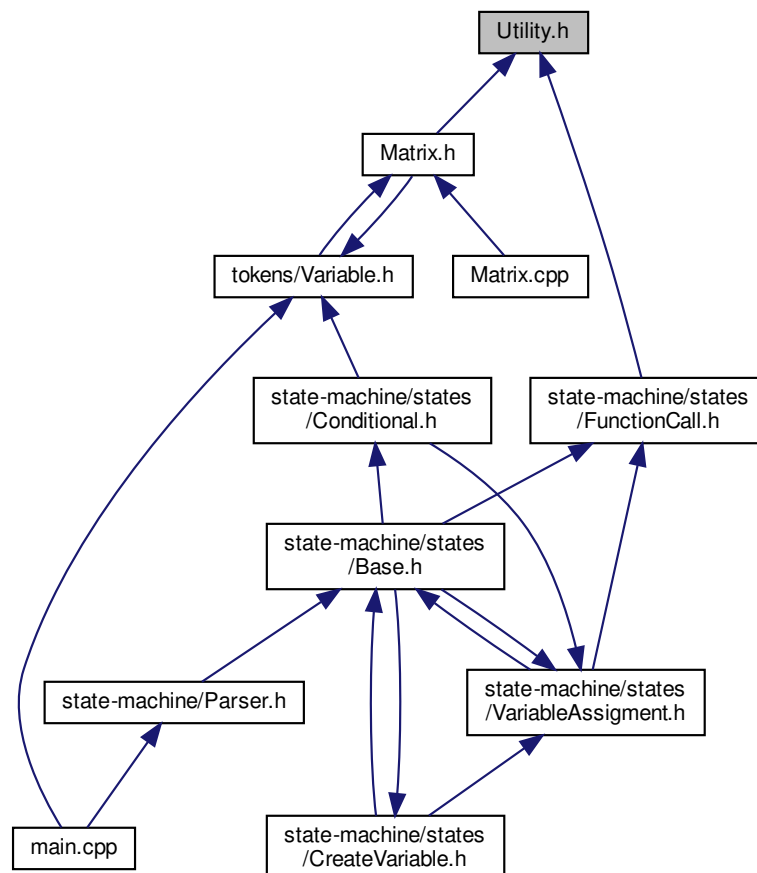
5.15.1 Typedef Documentation

5.15.1.1 ValueType

```
using ValueType = Matrix
```

5.16 Utility.h File Reference

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



Classes

- class [Utility](#)

Index

- [\\$NameState](#), [7](#)
 - [\\$NameState](#), [8](#)
 - [parse](#), [8](#)
- [\\$ValueState](#), [9](#)
 - [\\$ValueState](#), [9](#)
 - [parse](#), [10](#)
- [~FunctionCall](#)
 - [FunctionCall](#), [20](#)
- [add_column](#)
 - [Matrix](#), [22](#)
- [add_row](#)
 - [Matrix](#), [22](#)
- [add_value](#)
 - [Matrix](#), [23](#)
- [Base](#), [10](#)
 - [Base](#), [11](#)
 - [parse](#), [11](#)
- [CHANGE_STATE](#)
 - [State.h](#), [56](#)
- [Conditional](#), [12](#)
 - [Conditional](#), [13](#)
 - [parse](#), [13](#)
- [eq](#)
 - [main.cpp](#), [46](#)
- [Error](#), [14](#)
 - [Error](#), [14](#)
 - [parse](#), [15](#)
- [exit_func](#)
 - [main.cpp](#), [46](#)
- [find_token](#)
 - [Utility](#), [37](#)
- [FirstRowState](#), [15](#)
 - [FirstRowState](#), [16](#)
 - [parse](#), [16](#)
- [FUNCTION](#)
 - [Function.h](#), [64](#)
- [Function](#), [17](#)
 - [Function](#), [18](#)
 - [get_name](#), [18](#)
 - [get_type](#), [18](#)
 - [get_value](#), [19](#)
- [function](#)
 - [Token](#), [36](#)
- [Function.h](#)
 - [FUNCTION](#), [64](#)
- [FunctionCall](#), [19](#)
 - [~FunctionCall](#), [20](#)
 - [FunctionCall](#), [20](#)
 - [parse](#), [20](#)
- [get](#)
 - [Matrix](#), [23](#)
- [get_from_stack](#)
 - [Matrix](#), [23](#)
- [get_name](#)
 - [Function](#), [18](#)
 - [Scope](#), [32](#)
 - [Token](#), [36](#)
 - [Variable](#), [39](#)
- [get_type](#)
 - [Function](#), [18](#)
 - [Scope](#), [32](#)
 - [Token](#), [36](#)
 - [Variable](#), [39](#)
- [get_value](#)
 - [Function](#), [19](#)
 - [Variable](#), [40](#)
- [hello](#)
 - [main.cpp](#), [46](#)
- [input](#)
 - [main.cpp](#), [47](#)
- [is_matrix](#)
 - [Matrix](#), [24](#)
- [main](#)
 - [main.cpp](#), [47](#)
- [main.cpp](#), [45](#)
 - [eq](#), [46](#)
 - [exit_func](#), [46](#)
 - [hello](#), [46](#)
 - [input](#), [47](#)
 - [main](#), [47](#)
 - [newline](#), [48](#)
 - [not_func](#), [48](#)
 - [ones](#), [49](#)
 - [print](#), [50](#)
 - [text](#), [50](#)
- [Matrix](#), [21](#)
 - [add_column](#), [22](#)
 - [add_row](#), [22](#)
 - [add_value](#), [23](#)
 - [get](#), [23](#)
 - [get_from_stack](#), [23](#)
 - [is_matrix](#), [24](#)

- Matrix, 22
- operator(), 25
- operator==, 25
- parse_matrix, 25
- repr, 26
- size, 26
- translate, 27
- Matrix.cpp, 51
- Matrix.h, 52
- newline
 - main.cpp, 48
- not_func
 - main.cpp, 48
- ones
 - main.cpp, 49
- operator()
 - Matrix, 25
- operator==
 - Matrix, 25
- parse
 - \$NameState, 8
 - \$ValueState, 10
 - Base, 11
 - Conditional, 13
 - Error, 15
 - FirstRowState, 16
 - FunctionCall, 20
 - RowState, 30
 - State, 34
 - VariableAssignment, 42
- parse_matrix
 - Matrix, 25
- parse_string
 - Parser, 28
- Parser, 27
 - parse_string, 28
 - Parser, 28
 - stack_, 28
- print
 - main.cpp, 50
- repr
 - Matrix, 26
- RowState, 29
 - parse, 30
 - RowState, 30
- Scope, 31
 - get_name, 32
 - get_type, 32
 - Scope, 31
- scope
 - Token, 36
- set_value
 - Variable, 40
- size
 - Matrix, 26
- Stack
 - Token.h, 66
- stack_
 - Parser, 28
 - State, 34
- State, 33
 - parse, 34
 - stack_, 34
 - State, 33
- state-machine/Parser.h, 53
- state-machine/State.h, 54
- state-machine/states/Base.h, 56
- state-machine/states/Conditional.h, 57
- state-machine/states/CreateVariable.h, 58
- state-machine/states/Error.h, 60
- state-machine/states/FunctionCall.h, 60
- state-machine/states/VariableAssignment.h, 61
- State.h
 - CHANGE_STATE, 56
- text
 - main.cpp, 50
- Token, 35
 - function, 36
 - get_name, 36
 - get_type, 36
 - scope, 36
 - TokenType, 35
 - variable, 36
- Token.h
 - Stack, 66
- tokens/Function.h, 62
- tokens/Scope.h, 64
- tokens/Token.h, 66
- tokens/Variable.h, 67
- TokenType
 - Token, 35
- translate
 - Matrix, 27
- Utility, 36
 - find_token, 37
 - whitespace, 37
- Utility.h, 68
- ValueType
 - Variable.h, 68
- Variable, 38
 - get_name, 39
 - get_type, 39
 - get_value, 40
 - set_value, 40
 - Variable, 39
- variable
 - Token, 36
- Variable.h
 - ValueType, 68
- VariableAssignment, 41

parse, [42](#)

VariableAssignment, [42](#)

whitespace

Utility, [37](#)