YLang

Generated by Doxygen 1.9.1

1 Todo List 1

2 Namespace Index 3

2.1 Namespace List . . . 3

3 Hierarchical Index 5

3.1 Class Hierarchy . . . 5

4 Class Index 7

4.1 Class List . . . 7

5 File Index 9

5.1 File List . . . 9

6 Namespace Documentation 11

6.1 ylang Namespace Reference . . . 11

6.1.1 Detailed Description . . . 13

7 Class Documentation 15

7.1 ylang::AST Class Reference . . . 15

7.1.1 Detailed Description . . . 15

7.1.2 Member Function Documentation . . . 15

7.1.2.1 Print() . . . 15

7.2 ylang::ASTBinaryExpr Class Reference . . . 16

7.2.1 Detailed Description . . . 16

7.3 ylang::ASTBlockStmnt Class Reference . . . 16

7.3.1 Detailed Description . . . 17

7.4 ylang::ASTExpr Class Reference . . . 17

7.4.1 Detailed Description . . . 17

7.5 ylang::ASTExprStmnt Class Reference . . . 17

7.5.1 Detailed Description . . . 18

7.6 ylang::ASTGroupingExpr Class Reference . . . 18

7.6.1 Detailed Description . . . 18

7.7 ylang::ASTIfStmnt Class Reference . . . 19

7.7.1 Detailed Description . . . 19

7.8 ylang::ASTLiteral Class Reference . . . 19

7.8.1 Detailed Description . . . 20

7.9 ylang::ASTLogicalExpr Class Reference . . . 20

7.9.1 Detailed Description . . . 20

7.10 ylang::ASTNode Class Reference . . . 21

7.10.1 Detailed Description . . . 21

7.11 ylang::ASTPrinter Class Reference . . . 21

7.11.1 Detailed Description . . . 22

7.12 ylang::ASTPrintStmnt Class Reference . . . 22

7.12.1 Detailed Description . . . 22
7.13 ylang::ASTStmnt Class Reference . . . 22
7.13.1 Detailed Description . . . 23
7.14 ylang::ASTUnaryExpr Class Reference . . . 23
7.14.1 Detailed Description . . . 23
7.15 ylang::ASTVarAssignExpr Class Reference . . . 24
7.15.1 Detailed Description . . . 24
7.16 ylang::ASTVarExpr Class Reference . . . 24
7.16.1 Detailed Description . . . 25
7.17 ylang::ASTVarStmnt Class Reference . . . 25
7.17.1 Detailed Description . . . 25
7.18 ylang::ASTWhileStmnt Class Reference . . . 26
7.18.1 Detailed Description . . . 26
7.19 ylang::CmndLineArgs Class Reference . . . 26
7.19.1 Detailed Description . . . 27
7.19.2 Constructor & Destructor Documentation . . . 27
7.19.2.1 CmndLineArgs() [1/2] . . . 27
7.19.2.2 CmndLineArgs() [2/2] . . . 27
7.19.2.3 ~CmndLineArgs() . . . 28
7.19.3 Member Function Documentation . . . 28
7.19.3.1 ArgHasValue() . . . 28
7.19.3.2 ArgPresent() . . . 28
7.19.3.3 GetArgValue() . . . 29
7.19.4 Member Data Documentation . . . 29
7.19.4.1 args . . . 29
7.19.4.2 flags . . . 29
7.20 ylang::compiler_error Class Reference . . . 30
7.20.1 Detailed Description . . . 30
7.20.2 Constructor & Destructor Documentation . . . 30
7.20.2.1 compiler_error() [1/3] . . . 30
7.20.2.2 compiler_error() [2/3] . . . 31
7.20.2.3 compiler_error() [3/3] . . . 31
7.21 ylang::Error Struct Reference . . . 31
7.21.1 Detailed Description . . . 32
7.21.2 Constructor & Destructor Documentation . . . 32
7.21.2.1 Error() [1/8] . . . 32
7.21.2.2 Error() [2/8] . . . 32
7.21.2.3 Error() [3/8] . . . 33
7.21.2.4 Error() [4/8] . . . 33
7.21.2.5 Error() [5/8] . . . 33
7.21.2.6 Error() [6/8] . . . 34
7.21.2.7 Error() [7/8] . . . 34

7.21.2.8 Error() [8/8] . . . 34
7.21.3 Member Data Documentation . . . 34
7.21.3.1 level . . . 34
7.22 ylang::ErrorHandler Class Reference . . . 35
7.22.1 Detailed Description . . . 35
7.22.2 Member Function Documentation . . . 36
7.22.2.1 FlushErrors() . . . 36
7.22.2.2 HandleLexerError() . . . 36
7.22.2.3 HandleParserError() . . . 36
7.22.2.4 HandleRuntimeError() . . . 36
7.22.2.5 IsEmpty() . . . 36
7.22.2.6 SubmitError() . . . 36
7.23 ylang::ExprVisitor Class Reference . . . 37
7.23.1 Detailed Description . . . 37
7.24 ylang::Interpreter Class Reference . . . 37
7.24.1 Member Function Documentation . . . 38
7.24.1.1 VisitVarAssignExpr() . . . 38
7.25 ylang::interpreter_error Class Reference . . . 39
7.25.1 Detailed Description . . . 39
7.25.2 Constructor & Destructor Documentation . . . 39
7.25.2.1 interpreter_error() [1/3] . . . 39
7.25.2.2 interpreter_error() [2/3] . . . 40
7.25.2.3 interpreter_error() [3/3] . . . 40
7.26 ylang::Lexer Class Reference . . . 40
7.26.1 Detailed Description . . . 42
7.26.2 Constructor & Destructor Documentation . . . 42
7.26.2.1 Lexer() . . . 42
7.26.3 Member Function Documentation . . . 42
7.26.3.1 Advance() . . . 42
7.26.3.2 Consume() . . . 42
7.26.3.3 GetTokens() . . . 43
7.26.3.4 HandleComment() . . . 43
7.26.3.5 HandleString() . . . 43
7.26.3.6 InitTokenTypeMaps() . . . 43
7.26.3.7 Lex() . . . 43
7.26.3.8 NewLine() . . . 44
7.26.3.9 Peek() . . . 44
7.26.3.10 PeekNext() . . . 44
7.26.4 Member Data Documentation . . . 44
7.26.4.1 keywords . . . 44
7.26.4.2 operators . . . 45
7.27 ylang::lexer_error Class Reference . . . 45

7.27.1 Detailed Description . . . 45
7.27.2 Constructor & Destructor Documentation . . . 45
7.27.2.1 lexer_error() [1/3] . . . 45
7.27.2.2 lexer_error() [2/3] . . . 46
7.27.2.3 lexer_error() [3/3] . . . 46
7.28 ylang::Parser Class Reference . . . 47
7.29 ylang::parser_error Class Reference . . . 48
7.29.1 Detailed Description . . . 48
7.29.2 Constructor & Destructor Documentation . . . 48
7.29.2.1 parser_error() [1/3] . . . 48
7.29.2.2 parser_error() [2/3] . . . 49
7.29.2.3 parser_error() [3/3] . . . 49
7.30 ylang::Token Struct Reference . . . 49
7.30.1 Detailed Description . . . 50
7.31 ylang::Util Class Reference . . . 50
7.31.1 Detailed Description . . . 50
7.31.2 Member Function Documentation . . . 50
7.31.2.1 GetCurrFileName() . . . 51
7.31.2.2 PrintToken() . . . 51
7.31.2.3 PrintUsage() . . . 51
7.31.2.4 ReadSrc() . . . 51
7.32 ylang::Variable Struct Reference . . . 52
7.33 ylang::VM Struct Reference . . . 52

8 File Documentation **53**
8.1 YLang0.2/include/defines.hpp File Reference . . . 53
8.1.1 Detailed Description . . . 54
8.2 YLang0.2/include/errors.hpp File Reference . . . 55
8.2.1 Detailed Description . . . 55
8.3 YLang0.2/include/lexer.hpp File Reference . . . 56
8.4 YLang0.2/include/util.hpp File Reference . . . 56
8.4.1 Detailed Description . . . 56

Index **57**

Chapter 1

Todo List

Member ylang::ErrorHandler::HandleRuntimeError (const Error &error)

spit out stack trace

Member ylang::Interpreter::VisitVarAssignExpr (ASTVarAssignExpr &expr) override

Fix this, right now it only searches the current scope will fail if the variable is not in the current scope

Member ylang::Lexer::HandleComment ()

Handle comments

Member ylang::Lexer::HandleString ()

Handle string literals

Chapter 2

Namespace Index

2.1 Namespace List

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

ylang
All the ylang compiler related code 11

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

ylang::AST . . . 15
ylang::ASTNode . . . 21
  ylang::ASTExpr . . . 17
    ylang::ASTBinaryExpr . . . 16
    ylang::ASTGroupingExpr . . . 18
    ylang::ASTLiteral . . . 19
    ylang::ASTLogicalExpr . . . 20
    ylang::ASTUnaryExpr . . . 23
    ylang::ASTVarAssignExpr . . . 24
    ylang::ASTVarExpr . . . 24
  ylang::ASTStmnt . . . 22
    ylang::ASTBlockStmnt . . . 16
    ylang::ASTExprStmnt . . . 17
    ylang::ASTIfStmnt . . . 19
    ylang::ASTPrintStmnt . . . 22
    ylang::ASTVarStmnt . . . 25
    ylang::ASTWhileStmnt . . . 26
ylang::CmndLineArgs . . . 26
ylang::Error . . . 31
ylang::ErrorHandler . . . 35
ylang::ExprVisitor . . . 37
  ylang::ASTPrinter . . . 21
  ylang::Interpreter . . . 37
ylang::Lexer . . . 40
ylang::Parser . . . 47
std::runtime_error
  ylang::compiler_error . . . 30
  ylang::interpreter_error . . . 39
  ylang::lexer_error . . . 45
  ylang::parser_error . . . 48
ylang::Token . . . 49
ylang::Util . . . 50
ylang::Variable . . . 52
ylang::VM . . . 52

Chapter 4

Class Index

4.1 Class List

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

ylang::AST
Class used to represent the AST . . . 15
ylang::ASTBinaryExpr
Class used to represent a binary expression in the AST . . . 16
ylang::ASTBlockStmnt
Class used to represent a block statement in the AST . . . 16
ylang::ASTExpr
Abstract class used to represent an expression in the AST . . . 17
ylang::ASTExprStmnt
Class used to represent an expression statement in the AST . . . 17
ylang::ASTGroupingExpr
Class used to represent a grouping expression in the AST . . . 18
ylang::ASTIfStmnt
Class used to represent an if statement in the AST . . . 19
ylang::ASTLiteral
Class used to represent a literal in the AST . . . 19
ylang::ASTLogicalExpr
Class used to represent a logical expression in the AST . . . 20
ylang::ASTNode
Abstract class used to represent a node in the AST . . . 21
ylang::ASTPrinter
Class used to print the AST . . . 21
ylang::ASTPrintStmnt
Class used to represent a print statement in the AST . . . 22
ylang::ASTStmnt
Abstract class used to represent a statement in the AST . . . 22
ylang::ASTUnaryExpr
Class used to represent a unary expression in the AST . . . 23
ylang::ASTVarAssignExpr
Class used to represent a variable assignment expression in the AST . . . 24
ylang::ASTVarExpr
Class used to represent a variable expression in the AST . . . 24
ylang::ASTVarStmnt
Class used to represent a variable statement in the AST . . . 25
ylang::ASTWhileStmnt
Class used to represent a while statement in the AST . . . 26

ylang::CmndLineArgs
Used to parse the command line arguments . . . 26
ylang::compiler_error
Class used to throw compiler errors with line and column information with help to specify which part of the compiler the error message is coming from . . . 30
ylang::Error
Struct used to store errors in the error handler . . . 31
ylang::ErrorHandler
Class used to handle errors . . . 35
ylang::ExprVisitor
Abstract class used to visit a node in the AST . . . 37
ylang::Interpreter . . . 37
ylang::interpreter_error
Class used to throw compiler errors with line and column information with help to specify which part of the compiler the error message is coming from . . . 39
ylang::Lexer
Used internally by the ylang compiler to perform lexical analysis on the source code . . . 40
ylang::lexer_error
Class used to throw lexer errors with line and column information with help to specify which part of the compiler the error message is coming from . . . 45
ylang::Parser . . . 47
ylang::parser_error
Class used to throw parser errors with line and column information with help to specify which part of the compiler the error message is coming from . . . 48
ylang::Token . . . 49
ylang::Util
Used to read and validate the source code from a file as well as act as a store-all for miscallaneous utility functions . . . 50
ylang::Variable . . . 52
ylang::VM . . . 52

Chapter 5

File Index

5.1 File List

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

YLang0.2/include/**ast.hpp** . . . . . **??**
YLang0.2/include/defines.hpp
Main definitions and other core information 53
YLang0.2/include/errors.hpp
Major error handling code and classes including the error handler class and different error classes 55
YLang0.2/include/**interpreter.hpp** . . . . . **??**
YLang0.2/include/lexer.hpp
Lexer class header file, defines the interface for the Lexer class used internally by the ylang compiler 56
YLang0.2/include/**parser.hpp** . . . . . **??**
YLang0.2/include/util.hpp
Miscallaneous utility functions 56

Chapter 6

Namespace Documentation

6.1 ylang Namespace Reference

contains all the ylang compiler related code

Classes

- class ExprVisitor

  Abstract class used to visit a node in the AST.
- class ASTPrinter

  Class used to print the AST.
- class ASTNode

  Abstract class used to represent a node in the AST.
- class AST

  Class used to represent the AST.
- class ASTExpr

  Abstract class used to represent an expression in the AST.
- class ASTLiteral

  Class used to represent a literal in the AST.
- class ASTLogicalExpr

  Class used to represent a logical expression in the AST.
- class ASTUnaryExpr

  Class used to represent a unary expression in the AST.
- class ASTBinaryExpr

  Class used to represent a binary expression in the AST.
- class ASTGroupingExpr

  Class used to represent a grouping expression in the AST.
- class ASTVarExpr

  Class used to represent a variable expression in the AST.
- class ASTVarAssignExpr

  Class used to represent a variable assignment expression in the AST.
- class ASTStmnt

  Abstract class used to represent a statement in the AST.
- class ASTPrintStmnt

  Class used to represent a print statement in the AST.

- class ASTExprStmnt

  Class used to represent an expression statement in the AST.
- class ASTVarStmnt

  Class used to represent a variable statement in the AST.
- class ASTBlockStmnt

  Class used to represent a block statement in the AST.
- class ASTIfStmnt

  Class used to represent an if statement in the AST.
- class ASTWhileStmnt

  Class used to represent a while statement in the AST.
- struct Token
- class lexer_error

  Class used to throw lexer errors with line and column information with help to specify which part of the compiler the error message is coming from.
- class parser_error

  Class used to throw parser errors with line and column information with help to specify which part of the compiler the error message is coming from.
- class interpreter_error

  Class used to throw compiler errors with line and column information with help to specify which part of the compiler the error message is coming from.
- class compiler_error

  Class used to throw compiler errors with line and column information with help to specify which part of the compiler the error message is coming from.
- struct Error

  Struct used to store errors in the error handler.
- class ErrorHandler

  Class used to handle errors.
- struct Variable
- struct VM
- class Interpreter
- class Lexer

  used internally by the ylang compiler to perform lexical analysis on the source code
- class Parser
- class CmndLineArgs

  used to parse the command line arguments
- class Util

  used to read and validate the source code from a file as well as act as a store-all for miscallaneous utility functions

Typedefs

- using **VarValue** = std::variant< int8_t, int16_t, int32_t, int64_t, uint8_t, uint16_t, uint32_t, uint64_t, float, double, bool, char, std::string >

Enumerations

- enum class **Type** {
  START_OF_FILE , **KW_I8** , **KW_I16** , **KW_I32** ,
  KW_I64 , **KW_U8** , **KW_U16** , **KW_U32** ,
  KW_U64 , **KW_F32** , **KW_F64** , **KW_BOOL** ,
  KW_CHAR , **KW_STRING** , **KW_BYTE** , **KW_NULL** ,
  KW_VOID , **KW_FN** , **KW_IF** , **KW_ELSE** ,

KW_FOR , **KW_WHILE** , **KW_SWITCH** , **KW_RETURN** ,
KW_BREAK , **KW_CONTINUE** , **KW_TRUE** , **KW_FALSE** ,
KW_STRUCT , **KW_CLASS** , **KW_OBJECT** , **KW_ENUM** ,
KW_UNION , **KW_NAMESPACE** , **KW_USING** , **KW_IMPORT** ,
KW_EXPORT , **KW_PUBLIC** , **KW_PRIVATE** , **KW_INCLUDE** ,
KW_PRINT , **IDENTIFIER** , **OP_ADD** , **OP_SUB** ,
OP_MUL , **OP_DIV** , **OP_MOD** , **OP_OPEN_PAREN** ,
OP_CLOSE_PAREN , **OP_OPEN_BRACE** , **OP_CLOSE_BRACE** , **OP_OPEN_BRACKET** ,
OP_CLOSE_BRACKET , **OP_SEMICOLON** , **OP_COMMA** , **OP_DOT** ,
OP_COLON , **OP_DOLLAR** , **OP_HASH** , **OP_AT** ,
OP_EXCLAMATION , **OP_QUESTION** , **OP_AMPERSAND** , **OP_ASSIGN** ,
OP_LESS_THAN , **OP_GREATER_THAN** , **OP_BIT_OR** , **OP_BIT_AND** ,
OP_XOR , **OP_BIT_NOT** , **OP_ADD_ASSIGN** , **OP_SUB_ASSIGN** ,
OP_MUL_ASSIGN , **OP_DIV_ASSIGN** , **OP_MOD_ASSIGN** , **OP_EQUALS** ,
OP_NOT_EQUAL , **OP_LESS_THAN_EQUAL** , **OP_GREATER_THAN_EQUAL** , **OP_LOGICAL_AND** ,
OP_LOGICAL_OR , **OP_BIT_OR_ASSIGN** , **OP_BIT_AND_ASSIGN** , **OP_XOR_ASSIGN** ,
OP_BIT_NOT_ASSIGN , **OP_LEFT_SHIFT** , **OP_RIGHT_SHIFT** , **OP_LEFT_SHIFT_ASSIGN** ,
OP_RIGHT_SHIFT_ASSIGN , **OP_INCREMENT** , **OP_DECREMENT** , **OP_ARROW** ,
OP_SCOPE , **OP_CAPTURE** , **LIT_INT** , **LIT_HEX** ,
LIT_FLOAT , **LIT_CHAR** , **LIT_STRING** , **COMMENT** ,
ERROR_TOKEN , **END_OF_FILE** }

- enum class **ErrorLevel** { **WARNING** , **FATAL** , **UNDEFINED_BEHAVIOR** }
- enum class **VarType** : uint8_t {
  I8 , **I16** , **I32** , **I64** ,
  U8 , **U16** , **U32** , **U64** ,
  F32 , **F64** , **BOOL** , **CHAR** ,
  STRING , **BYTE** , **NULL_** , **VOID** ,
  STRUCT , **CLASS** , **OBJECT** , **ENUM** ,
  UNION }

Functions

- VarValue **operator+** (const VarValue &lhs, const VarValue &rhs)
- VarValue **operator-** (const VarValue &lhs, const VarValue &rhs)
- VarValue **operator**∗ (const VarValue &lhs, const VarValue &rhs)
- VarValue **operator**/ (const VarValue &lhs, const VarValue &rhs)
- VarValue **operator%** (const VarValue &lhs, const VarValue &rhs)
- int **Main** (const std::string &src)

Variables

- static const std::string **YLANG_VERSION** = "0.0.2"
- static const std::string **TokenTypeStrings** [ ]

6.1.1 Detailed Description

contains all the ylang compiler related code

namespace

Chapter 7

Class Documentation

7.1 ylang::AST Class Reference

Class used to represent the AST.

`#include <ast.hpp>`

Public Member Functions

- void Print () const
  function used to print the AST

Public Attributes

- std::shared_ptr< ASTNode > **root**

7.1.1 Detailed Description

Class used to represent the AST.

7.1.2 Member Function Documentation

7.1.2.1 Print()

`void ylang::AST::Print ( ) const`

function used to print the AST

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.2 ylang::ASTBinaryExpr Class Reference

Class used to represent a binary expression in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTBinaryExpr:

Collaboration diagram for ylang::ASTBinaryExpr:

Public Member Functions

- **ASTBinaryExpr** (std::shared_ptr< ASTExpr > lhs, Token op, std::shared_ptr< ASTExpr > rhs)
- virtual void **Accept** (ExprVisitor &visitor) override
- virtual void **Print** () const override
- virtual Type **GetType** () const override

Public Attributes

- std::shared_ptr< ASTExpr > **lhs**
- Token **op**
- std::shared_ptr< ASTExpr > **rhs**

7.2.1 Detailed Description

Class used to represent a binary expression in the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.3 ylang::ASTBlockStmnt Class Reference

Class used to represent a block statement in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTBlockStmnt:

Collaboration diagram for ylang::ASTBlockStmnt:

Public Member Functions

- **ASTBlockStmnt** (std::vector< std::shared_ptr< ASTStmnt >> statements)
- virtual void **Accept** (ExprVisitor &visitor) override
- virtual void **Print** () const override

Public Attributes

- std::vector< std::shared_ptr< ASTStmnt > > **statements**

7.3.1 Detailed Description

Class used to represent a block statement in the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.4 ylang::ASTExpr Class Reference

Abstract class used to represent an expression in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTExpr:

Collaboration diagram for ylang::ASTExpr:

Public Member Functions

- virtual void **Accept** (ExprVisitor &visitor)=0
- virtual void **Print** () const =0
- virtual Type **GetType** () const =0

7.4.1 Detailed Description

Abstract class used to represent an expression in the AST.

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

- YLang0.2/include/ast.hpp

7.5 ylang::ASTExprStmnt Class Reference

Class used to represent an expression statement in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTExprStmnt:

Collaboration diagram for ylang::ASTExprStmnt:

Public Member Functions

- **ASTExprStmnt** (std::shared_ptr< ASTExpr > expr)
- virtual void **Accept** (ExprVisitor &visitor) override
- virtual void **Print** () const override

Public Attributes

- std::shared_ptr< ASTExpr > **expr**

7.5.1 Detailed Description

Class used to represent an expression statement in the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.6 ylang::ASTGroupingExpr Class Reference

Class used to represent a grouping expression in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTGroupingExpr:

Collaboration diagram for ylang::ASTGroupingExpr:

Public Member Functions

- **ASTGroupingExpr** (std::shared_ptr< ASTExpr > expr)
- virtual void **Accept** (ExprVisitor &visitor) override
- virtual void **Print** () const override
- virtual Type **GetType** () const override

Public Attributes

- std::shared_ptr< ASTExpr > **expr**

7.6.1 Detailed Description

Class used to represent a grouping expression in the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.7 ylang::ASTIfStmnt Class Reference

Class used to represent an if statement in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTIfStmnt:

Collaboration diagram for ylang::ASTIfStmnt:

Public Member Functions

- **ASTIfStmnt** (std::shared_ptr< ASTExpr > condition, std::shared_ptr< ASTStmnt > then_branch, std::shared_ptr< ASTStmnt > else_branch)
- virtual void **Accept** (ExprVisitor &visitor) override
- virtual void **Print** () const override

Public Attributes

- std::shared_ptr< ASTExpr > **condition**
- std::shared_ptr< ASTStmnt > **then_branch**
- std::shared_ptr< ASTStmnt > **else_branch**

7.7.1 Detailed Description

Class used to represent an if statement in the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.8 ylang::ASTLiteral Class Reference

Class used to represent a literal in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTLiteral:

Collaboration diagram for ylang::ASTLiteral:

Public Member Functions

- **ASTLiteral** (Type type, Token literal)
- virtual void **Accept** (ExprVisitor &visitor) override
- virtual void **Print** () const override
- virtual Type **GetType** () const override

Public Attributes

- Type **type**
- Token **literal**

7.8.1 Detailed Description

Class used to represent a literal in the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.9 ylang::ASTLogicalExpr Class Reference

Class used to represent a logical expression in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTLogicalExpr:

Collaboration diagram for ylang::ASTLogicalExpr:

Public Member Functions

- **ASTLogicalExpr** (std::shared_ptr< ASTExpr > lhs, Token op, std::shared_ptr< ASTExpr > rhs)
- virtual void **Accept** (ExprVisitor &visitor) override
- virtual void **Print** () const override
- virtual Type **GetType** () const override

Public Attributes

- std::shared_ptr< ASTExpr > **lhs**
- Token **op**
- std::shared_ptr< ASTExpr > **rhs**

7.9.1 Detailed Description

Class used to represent a logical expression in the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.10 ylang::ASTNode Class Reference

Abstract class used to represent a node in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTNode:

Public Member Functions

- virtual void **Accept** (ExprVisitor &visitor)=0
- virtual void **Print** () const =0

7.10.1 Detailed Description

Abstract class used to represent a node in the AST.

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

- YLang0.2/include/ast.hpp

7.11 ylang::ASTPrinter Class Reference

Class used to print the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTPrinter:

Collaboration diagram for ylang::ASTPrinter:

Public Member Functions

- virtual void **VisitLiteralExpr** (ASTLiteral &expr) override
- virtual void **VisitLogicalExpr** (ASTLogicalExpr &expr) override
- virtual void **VisitUnaryExpr** (ASTUnaryExpr &expr) override
- virtual void **VisitBinaryExpr** (ASTBinaryExpr &expr) override
- virtual void **VisitGroupingExpr** (ASTGroupingExpr &expr) override
- virtual void **VisitVarExpr** (ASTVarExpr &expr) override
- virtual void **VisitVarAssignExpr** (ASTVarAssignExpr &expr) override
- virtual void **VisitPrintStmnt** (ASTPrintStmnt &stmnt) override
- virtual void **VisitExprStmnt** (ASTExprStmnt &stmnt) override
- virtual void **VisitVarStmnt** (ASTVarStmnt &stmnt) override
- virtual void **VisitBlockStmnt** (ASTBlockStmnt &stmnt) override
- virtual void **VisitIfStmnt** (ASTIfStmnt &stmnt) override
- virtual void **VisitWhileStmnt** (ASTWhileStmnt &stmnt) override

7.11.1 Detailed Description

Class used to print the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.12 ylang::ASTPrintStmnt Class Reference

Class used to represent a print statement in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTPrintStmnt:

Collaboration diagram for ylang::ASTPrintStmnt:

Public Member Functions

- **ASTPrintStmnt** (std::shared_ptr< ASTExpr > expr)
- virtual void **Accept** (ExprVisitor &visitor) override
- virtual void **Print** () const override

Public Attributes

- std::shared_ptr< ASTExpr > **expr**

7.12.1 Detailed Description

Class used to represent a print statement in the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.13 ylang::ASTStmnt Class Reference

Abstract class used to represent a statement in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTStmnt:

Collaboration diagram for ylang::ASTStmnt:

Public Member Functions

- virtual void **Accept** (ExprVisitor &visitor)=0
- virtual void **Print** () const =0

7.13.1 Detailed Description

Abstract class used to represent a statement in the AST.

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

- YLang0.2/include/ast.hpp

7.14 ylang::ASTUnaryExpr Class Reference

Class used to represent a unary expression in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTUnaryExpr:

Collaboration diagram for ylang::ASTUnaryExpr:

Public Member Functions

- **ASTUnaryExpr** (Token op, std::shared_ptr< ASTExpr > expr)
- virtual void **Accept** (ExprVisitor &visitor) override
- virtual void **Print** () const override
- virtual Type **GetType** () const override

Public Attributes

- Token **op**
- std::shared_ptr< ASTExpr > **expr**

7.14.1 Detailed Description

Class used to represent a unary expression in the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.15 ylang::ASTVarAssignExpr Class Reference

Class used to represent a variable assignment expression in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTVarAssignExpr:

Collaboration diagram for ylang::ASTVarAssignExpr:

Public Member Functions

- **ASTVarAssignExpr** (Token name, std::shared_ptr< ASTExpr > assignment)
- virtual void **Accept** (ExprVisitor &visitor) override
- virtual void **Print** () const override
- virtual Type **GetType** () const override

Public Attributes

- Token **name**
- std::shared_ptr< ASTExpr > **assignment**

7.15.1 Detailed Description

Class used to represent a variable assignment expression in the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.16 ylang::ASTVarExpr Class Reference

Class used to represent a variable expression in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTVarExpr:

Collaboration diagram for ylang::ASTVarExpr:

Public Member Functions

- **ASTVarExpr** (Token name)
- virtual void **Accept** (ExprVisitor &visitor) override
- virtual void **Print** () const override
- virtual Type **GetType** () const override

Public Attributes

- Token **name**

7.16.1 Detailed Description

Class used to represent a variable expression in the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.17 ylang::ASTVarStmnt Class Reference

Class used to represent a variable statement in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTVarStmnt:

Collaboration diagram for ylang::ASTVarStmnt:

Public Member Functions

- **ASTVarStmnt** (Token type, Token name, std::shared_ptr< ASTExpr > initializer)
- virtual void **Accept** (ExprVisitor &visitor) override
- virtual void **Print** () const override

Public Attributes

- Token **type**
- Token **name**
- std::shared_ptr< ASTExpr > **initializer**

7.17.1 Detailed Description

Class used to represent a variable statement in the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.18 ylang::ASTWhileStmnt Class Reference

Class used to represent a while statement in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ASTWhileStmnt:

Collaboration diagram for ylang::ASTWhileStmnt:

Public Member Functions

- **ASTWhileStmnt** (std::shared_ptr< ASTExpr > condition, std::shared_ptr< ASTStmnt > body)
- virtual void **Accept** (ExprVisitor &visitor) override
- virtual void **Print** () const override

Public Attributes

- std::shared_ptr< ASTExpr > **condition**
- std::shared_ptr< ASTStmnt > **body**

7.18.1 Detailed Description

Class used to represent a while statement in the AST.

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

- YLang0.2/include/ast.hpp
- YLang0.2/src/ast.cpp

7.19 ylang::CmndLineArgs Class Reference

used to parse the command line arguments

`#include <util.hpp>`

Public Member Functions

- CmndLineArgs (int *&argc, char *argv[])
  parses the command line arguments
- CmndLineArgs ()=delete
- **CmndLineArgs** (const CmndLineArgs &)=delete
- **CmndLineArgs** (CmndLineArgs &&)=delete
- CmndLineArgs & **operator=** (const CmndLineArgs &)=delete
- CmndLineArgs & **operator=** (CmndLineArgs &&)=delete
- ~CmndLineArgs ()
  destructor (defualt)
- const bool ArgPresent (char flag) const
- const bool ArgHasValue (char flag) const
- const std::string GetArgValue (char flag) const

Private Member Functions

- char **GetFlagFromString** (const std::string &flag) const
- const bool **FlagRequiresValue** (char flag) const

Private Attributes

- std::map< char, std::optional< std::string > > args {}

  list of all the command line arguments

Static Private Attributes

- static std::vector< std::string > flags

  list of all the flags

Friends

- class **Util**

7.19.1 Detailed Description

used to parse the command line arguments

7.19.2 Constructor & Destructor Documentation

7.19.2.1 CmndLineArgs() [1/2]

```
ylang::CmndLineArgs::CmndLineArgs (
            int *& argc,
            char * argv[] )
```

parses the command line arguments

Parameters

argc	number of command line arguments
argv	list of command line arguments

7.19.2.2 CmndLineArgs() [2/2]

```
ylang::CmndLineArgs::CmndLineArgs ( )  [delete]
```

Note

we delete these to prevent copying and moving and ensure we only have one instance of our arugments and control data

7.19.2.3 ~CmndLineArgs()

```
ylang::CmndLineArgs::~CmndLineArgs ( )  [inline]
```

destructor (defualt)

7.19.3 Member Function Documentation

7.19.3.1 ArgHasValue()

```
const bool ylang::CmndLineArgs::ArgHasValue (
            char flag ) const  [inline]
```

Parameters

flag	flag of the command line argument

Returns

bool : true if the command line argument has a value, false otherwise

7.19.3.2 ArgPresent()

```
const bool ylang::CmndLineArgs::ArgPresent (
            char flag ) const  [inline]
```

Parameters

flag	Checks if flag is present in the command line arguments

Returns

bool : true if the command line argument is present, false otherwise

7.19.3.3 GetArgValue()

```
const std::string ylang::CmndLineArgs::GetArgValue (
            char flag ) const  [inline]
```

Parameters

flag	flag of the command line argument

Returns

std::string value of the command line argument

Exceptions

runtime	error if the command line argument is not present

7.19.4 Member Data Documentation

7.19.4.1 args

```
std::map<char , std::optional<std::string> > ylang::CmndLineArgs::args {}  [private]
```

list of all the command line arguments

7.19.4.2 flags

```
std::vector< std::string > ylang::CmndLineArgs::flags  [static], [private]
```

Initial value:

```
= {
        "-h" , " --help" ,
        "-v" , " --version" ,
        "-d" , " --debug" ,
        "-D" , " --dev" ,
        "-o" ,
        "-t" , "--time" ,
        "-l" ,
        "-a" ,
        "-S" , "--asm-only" ,
        "-O" , "--optimize" ,
    }
```

list of all the flags

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

- YLang0.2/include/util.hpp
- YLang0.2/src/util.cpp

7.20 ylang::compiler_error Class Reference

Class used to throw compiler errors with line and column information with help to specify which part of the compiler the error message is coming from.

`#include <errors.hpp>`

Inheritance diagram for ylang::compiler_error:

Collaboration diagram for ylang::compiler_error:

Public Member Functions

- compiler_error (const std::string &msg)
- compiler_error (uint32_t line, uint32_t col, const std::string &msg)
- compiler_error (ErrorLevel level, uint32_t line, uint32_t col, const std::string &msg)

Public Attributes

- ErrorLevel **level** = ErrorLevel::UNDEFINED_BEHAVIOR
- uint32_t **line** = 0
- uint32_t **column** = 0

7.20.1 Detailed Description

Class used to throw compiler errors with line and column information with help to specify which part of the compiler the error message is coming from.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 compiler_error() [1/3]

```
ylang::compiler_error::compiler_error (
            const std::string & msg )  [inline]
```

Parameters

msg	the error message

Note

This constructor is used for debugging purposes only and helps determine if the program is behaving as expected

7.20.2.2 compiler_error() [2/3]

```
ylang::compiler_error::compiler_error (
            uint32_t line,
            uint32_t col,
            const std::string & msg )  [inline]
```

Parameters

line	the line number of the error
col	the column number of the error
msg	the error message

Note

This constructor is used for debugging purposes only and helps determine if the program is behaving as expected

7.20.2.3 compiler_error() [3/3]

```
ylang::compiler_error::compiler_error (
            ErrorLevel level,
            uint32_t line,
            uint32_t col,
            const std::string & msg )  [inline]
```

Parameters

level	the level of the error
line	the line number of the error
col	the column number of the error
msg	the error message

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

- YLang0.2/include/errors.hpp

7.21 ylang::Error Struct Reference

Struct used to store errors in the error handler.

```
#include <errors.hpp>
```

Public Member Functions

- Error ()
- Error (const std::string &msg)
- Error (uint32_t line, uint32_t col, const std::string &msg)
- Error (ErrorLevel level, uint32_t line, uint32_t col, const std::string &msg)
- Error (const lexer_error &error)
- Error (const parser_error &error)
- Error (const interpreter_error &error)
- Error (const compiler_error &error)

Public Attributes

- ErrorLevel level = ErrorLevel::UNDEFINED_BEHAVIOR
- uint32_t **line** = 0
- uint32_t **column** = 0
- std::string **msg** = ""

7.21.1 Detailed Description

Struct used to store errors in the error handler.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 Error() [1/8]

```
ylang::Error::Error ( )  [inline]
```

Note

This constructor is used for debugging purposes only and helps determine if the program is behaving as expected

7.21.2.2 Error() [2/8]

```
ylang::Error::Error (
            const std::string & msg )  [inline]
```

Parameters

msg	the error message

Note

This constructor is used for debugging purposes only and helps determine if the program is behaving as expected

7.21.2.3 Error() [3/8]

```
ylang::Error::Error (
            uint32_t line,
            uint32_t col,
            const std::string & msg )  [inline]
```

Parameters

line	the line number of the error
col	the column number of the error
msg	the error message

Note

This constructor is the recommended constructor to use when throwing errors

7.21.2.4 Error() [4/8]

```
ylang::Error::Error (
            ErrorLevel level,
            uint32_t line,
            uint32_t col,
            const std::string & msg )  [inline]
```

Parameters

level	the level of the error
line	the line number of the error
column	the column number of the error
msg	the error message

7.21.2.5 Error() [5/8]

```
ylang::Error::Error (
            const lexer_error & error )
```

Parameters

error	the ylang lexer error

7.21.2.6 Error() [6/8]

```
ylang::Error::Error (
            const parser_error & error )
```

Parameters

error	the ylang parser error

7.21.2.7 Error() [7/8]

```
ylang::Error::Error (
            const interpreter_error & error )
```

Parameters

error	the ylang interpreter error

7.21.2.8 Error() [8/8]

```
ylang::Error::Error (
            const compiler_error & error )
```

Parameters

error	the ylang compiler error

7.21.3 Member Data Documentation

7.21.3.1 level

```
ErrorLevel ylang::Error::level = ErrorLevel::UNDEFINED_BEHAVIOR
```

Note

we initialize the level to undefined behavior for the sake of safety

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

- YLang0.2/include/errors.hpp
- YLang0.2/src/errors.cpp

7.22 ylang::ErrorHandler Class Reference

Class used to handle errors.

`#include <errors.hpp>`

Public Member Functions

- void SubmitError (const Error &error)

  Add an error to the error queue.
- bool IsEmpty () const

  Check if the error queue is empty.
- void FlushErrors ()

  Flush the error queue if parse was aborted.
- void **ClearErrors** ()
- void HandleLexerError (const Error &error, const std::string &src, uint32_t line_strt_index, uint32_t index)

  Formats and prints lexer error.
- void HandleParserError (const Error &error)

  Formats and prints parser error.
- void HandleRuntimeError (const Error &error)

  Formats and prints interpreter error.

Private Member Functions

- std::string **FormatLexerError** (const Error &error, const std::string &src, uint32_t line_strt_index, uint32_t index)
- std::string **FormatError** (const Error &err)

Private Attributes

- std::stack< Error > **error_queue**

7.22.1 Detailed Description

Class used to handle errors.

7.22.2 Member Function Documentation

7.22.2.1 FlushErrors()

```
void ylang::ErrorHandler::FlushErrors ( )
```

Flush the error queue if parse was aborted.

7.22.2.2 HandleLexerError()

```
void ylang::ErrorHandler::HandleLexerError (
            const Error & error,
            const std::string & src,
            uint32_t line_strt_index,
            uint32_t index )
```

Formats and prints lexer error.

7.22.2.3 HandleParserError()

```
void ylang::ErrorHandler::HandleParserError (
            const Error & error )
```

Formats and prints parser error.

7.22.2.4 HandleRuntimeError()

```
void ylang::ErrorHandler::HandleRuntimeError (
            const Error & error )
```

Formats and prints interpreter error.

Todo spit out stack trace

7.22.2.5 IsEmpty()

```
bool ylang::ErrorHandler::IsEmpty ( ) const  [inline]
```

Check if the error queue is empty.

Returns

bool : true if the error queue is empty, false otherwise

7.22.2.6 SubmitError()

```
void ylang::ErrorHandler::SubmitError (
            const Error & error )  [inline]
```

Add an error to the error queue.

Parameters

error	the error to add

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

- YLang0.2/include/errors.hpp
- YLang0.2/src/errors.cpp

7.23 ylang::ExprVisitor Class Reference

Abstract class used to visit a node in the AST.

`#include <ast.hpp>`

Inheritance diagram for ylang::ExprVisitor:

Public Member Functions

- virtual void **VisitLiteralExpr** (ASTLiteral &expr)=0
- virtual void **VisitLogicalExpr** (ASTLogicalExpr &expr)=0
- virtual void **VisitUnaryExpr** (ASTUnaryExpr &expr)=0
- virtual void **VisitBinaryExpr** (ASTBinaryExpr &expr)=0
- virtual void **VisitGroupingExpr** (ASTGroupingExpr &expr)=0
- virtual void **VisitVarExpr** (ASTVarExpr &expr)=0
- virtual void **VisitVarAssignExpr** (ASTVarAssignExpr &expr)=0
- virtual void **VisitPrintStmnt** (ASTPrintStmnt &stmnt)=0
- virtual void **VisitExprStmnt** (ASTExprStmnt &stmnt)=0
- virtual void **VisitVarStmnt** (ASTVarStmnt &stmnt)=0
- virtual void **VisitBlockStmnt** (ASTBlockStmnt &stmnt)=0
- virtual void **VisitIfStmnt** (ASTIfStmnt &stmnt)=0
- virtual void **VisitWhileStmnt** (ASTWhileStmnt &stmnt)=0

7.23.1 Detailed Description

Abstract class used to visit a node in the AST.

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

- YLang0.2/include/ast.hpp

7.24 ylang::Interpreter Class Reference

Inheritance diagram for ylang::Interpreter:

Collaboration diagram for ylang::Interpreter:

Public Member Functions

- virtual void **VisitLiteralExpr** (ASTLiteral &expr) override
- virtual void **VisitLogicalExpr** (ASTLogicalExpr &expr) override
- virtual void **VisitUnaryExpr** (ASTUnaryExpr &expr) override
- virtual void **VisitBinaryExpr** (ASTBinaryExpr &expr) override
- virtual void **VisitGroupingExpr** (ASTGroupingExpr &expr) override
- virtual void **VisitVarExpr** (ASTVarExpr &expr) override
- virtual void VisitVarAssignExpr (ASTVarAssignExpr &expr) override
- virtual void **VisitPrintStmnt** (ASTPrintStmnt &stmnt) override
- virtual void **VisitExprStmnt** (ASTExprStmnt &stmnt) override
- virtual void **VisitVarStmnt** (ASTVarStmnt &stmnt) override
- virtual void **VisitBlockStmnt** (ASTBlockStmnt &stmnt) override
- virtual void **VisitIfStmnt** (ASTIfStmnt &stmnt) override
- virtual void **VisitWhileStmnt** (ASTWhileStmnt &stmnt) override

Private Member Functions

- Variable **MakeVar** (VarType type, const VarValue &val)
- Variable * **GetVar** (const std::string &name)
- void **PushEnv** ()
- void **PopEnv** ()
- void **EvaluateBinaryExpr** (Type op, const Variable &lhs, const Variable &rhs)
- bool **IsTruthy** (const Variable &val)

7.24.1 Member Function Documentation

7.24.1.1 VisitVarAssignExpr()

```
void ylang::Interpreter::VisitVarAssignExpr (
            ASTVarAssignExpr & expr )  [override], [virtual]
```

Todo Fix this, right now it only searches the current scope will fail if the variable is not in the current scope

Note

Need to change variable in place in environment it already exists in

Implements ylang::ExprVisitor.

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

- YLang0.2/include/interpreter.hpp
- YLang0.2/src/interpreter.cpp

7.25 ylang::interpreter_error Class Reference

Class used to throw compiler errors with line and column information with help to specify which part of the compiler the error message is coming from.

`#include <errors.hpp>`

Inheritance diagram for ylang::interpreter_error:

Collaboration diagram for ylang::interpreter_error:

Public Member Functions

- interpreter_error (const std::string &msg)
- interpreter_error (uint32_t line, uint32_t col, const std::string &msg)
- interpreter_error (ErrorLevel level, uint32_t line, uint32_t col, const std::string &msg)

Public Attributes

- ErrorLevel **level** = ErrorLevel::UNDEFINED_BEHAVIOR
- uint32_t **line** = 0
- uint32_t **column** = 0

7.25.1 Detailed Description

Class used to throw compiler errors with line and column information with help to specify which part of the compiler the error message is coming from.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 interpreter_error() [1/3]

```
ylang::interpreter_error::interpreter_error (
            const std::string & msg )  [inline]
```

Parameters

msg	the error message

Note

This constructor is used for debugging purposes only and helps determine if the program is behaving as expected

7.25.2.2 interpreter_error() [2/3]

```
ylang::interpreter_error::interpreter_error (
            uint32_t line,
            uint32_t col,
            const std::string & msg )  [inline]
```

Parameters

line	the line number of the error
col	the column number of the error
msg	the error message

Note

This constructor is used for debugging purposes only and helps determine if the program is behaving as expected

7.25.2.3 interpreter_error() [3/3]

```
ylang::interpreter_error::interpreter_error (
            ErrorLevel level,
            uint32_t line,
            uint32_t col,
            const std::string & msg )  [inline]
```

Parameters

level	the level of the error
line	the line number of the error
col	the column number of the error
msg	the error message

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

- YLang0.2/include/errors.hpp

7.26 ylang::Lexer Class Reference

used internally by the ylang compiler to perform lexical analysis on the source code

`#include <lexer.hpp>`

Public Member Functions

- Lexer (std::shared_ptr< ErrorHandler > err_handler, const std::string &src)
- bool Lex ()

  Method called to lex the source code.
- const std::vector< Token > GetTokens () const

Private Member Functions

- bool **IsNumeric** (char c)
- bool **IsAlpha** (char c)
- bool **IsAlphaNumeric** (char c)
- bool **IsOperator** (char c) const
- bool **IsWhitespace** (char c)
- bool **IsKeyword** (const std::string &str) const
- Type **GetTypeForKeyword** (const std::string &str) const
- Type **GetTypeForIdentifier** (const std::string &str) const
- void NewLine (bool iterate_src_index=true)

  iterates src if flag is set, infrements line and sets column to 1, also sets error helper variables to the correct values
- void Consume ()

  increments to next character without saving to current token
- void Advance ()

  saves to current token and increments to next character
- void **DiscardToken** ()
- void **AddToken** (Type type)
- char Peek () const

  looks at current character while also checking bounds
- char PeekNext () const

  looks at next character while also checking bounds
- bool **Check** (char c) const
- bool **Match** (char c)
- void InitTokenTypeMaps ()

  fills map with strings corresponfing ylang::Type to identify token types
- void **HandleWhitespace** ()
- void HandleComment ()
- void **HandleChar** ()
- void HandleString ()
- void **HandleNumeric** ()
- void **HandleAlphaNumeric** ()
- void **HandleOperator** ()
- bool **IsEOF** () const

Private Attributes

- const std::vector< char > **operators**
- const std::vector< std::string > **keywords**
- std::map< std::string, Type > **keyword_map**
- std::map< std::string, Type > **identifier_map**
- std::shared_ptr< ErrorHandler > **error_handler** = nullptr
- uint32_t **start_of_line_index** = 0
- uint32_t **src_index** = 0
- uint32_t **src_length** = 0
- uint32_t **line** = 1
- uint32_t **column** = 1
- std::string **curr_token** = ""
- std::string **source** = "<n>"
- std::vector< Token > **tokens** {}

7.26.1 Detailed Description

used internally by the ylang compiler to perform lexical analysis on the source code

Note

this class is not meant to be used by the end user

7.26.2 Constructor & Destructor Documentation

7.26.2.1 Lexer()

```
ylang::Lexer::Lexer (
            std::shared_ptr< ErrorHandler > err_handler,
            const std::string & src )  [inline]
```

Parameters

err_handler	pointer to the error handler
src	source code to lex

7.26.3 Member Function Documentation

7.26.3.1 Advance()

```
void ylang::Lexer::Advance ( )  [inline], [private]
```

saves to current token and increments to next character

7.26.3.2 Consume()

```
void ylang::Lexer::Consume ( )  [inline], [private]
```

increments to next character without saving to current token

7.26.3.3 GetTokens()

```
const std::vector<Token> ylang::Lexer::GetTokens ( ) const  [inline]
```

Returns

std::vector<Token> vector of tokens

7.26.3.4 HandleComment()

```
void ylang::Lexer::HandleComment ( )  [private]
```

Todo Handle comments

7.26.3.5 HandleString()

```
void ylang::Lexer::HandleString ( )  [private]
```

Todo Handle string literals

7.26.3.6 InitTokenTypeMaps()

```
void ylang::Lexer::InitTokenTypeMaps ( )  [private]
```

fills map with strings corresponfing ylang::Type to identify token types

7.26.3.7 Lex()

```
bool ylang::Lexer::Lex ( )
```

Method called to lex the source code.

Exceptions

ylang_lexer_error	if unhandleable error is encountered

7.26.3.8 NewLine()

```
void ylang::Lexer::NewLine (
            bool iterate_src_index = true )  [inline], [private]
```

iterates src if flag is set, infrements line and sets column to 1, also sets error helper variables to the correct values

Parameters

iterate_src_index	(optional, default = true)

7.26.3.9 Peek()

```
char ylang::Lexer::Peek ( ) const  [inline], [private]
```

looks at current character while also checking bounds

7.26.3.10 PeekNext()

```
char ylang::Lexer::PeekNext ( ) const  [inline], [private]
```

looks at next character while also checking bounds

7.26.4 Member Data Documentation

7.26.4.1 keywords

```
const std::vector<std::string> ylang::Lexer::keywords  [private]
```

Initial value:

```
= {

            "i8"      , "i16"     , "i32"    , "i64"      ,
            "u8"      , "u16"     , "u32"    , "u64"      ,
            "f32"     , "f64"     , "double" , "bool"     ,
            "char"    , "string"  , "byte"   ,
            "null"   , "void"     ,

            "fn"      ,
            "if"      , "else"    , "for"    , "while"    ,
            "switch"  , "return"  , "break"  , "continue" ,
            "true"    , "false"   ,

            "struct"  , "class"   , "object" , "enum"     ,
            "union"   ,


            "namespace",
            "using"   , "import"  , "export" , "public"   ,
            "private" , "include" ,

            "print"
        }
```

7.26.4.2 operators

```
const std::vector<char> ylang::Lexer::operators  [private]
```

Initial value:

```
= {
            '+' , '-' , '*' , '/' , '%' ,
            '(' , ')' , '{' , '}' , '[' , ']' ,
            ';' , ',' , '.' , '"' , '\"' , ':' ,
            '$' , '#' , '@' , '!' , '?' , '&' ,
            '|' , '^' , '~' , '=' , '<' , '>'
        }
```

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

- YLang0.2/include/lexer.hpp
- YLang0.2/src/lexer.cpp

7.27 ylang::lexer_error Class Reference

Class used to throw lexer errors with line and column information with help to specify which part of the compiler the error message is coming from.

`#include <errors.hpp>`

Inheritance diagram for ylang::lexer_error:

Collaboration diagram for ylang::lexer_error:

Public Member Functions

- lexer_error (const std::string &msg)
- lexer_error (uint32_t line, uint32_t col, const std::string &msg)
- lexer_error (ErrorLevel level, uint32_t line, uint32_t col, const std::string &msg)

Public Attributes

- ErrorLevel **level** = ErrorLevel::UNDEFINED_BEHAVIOR
- uint32_t **line** = 0
- uint32_t **column** = 0

7.27.1 Detailed Description

Class used to throw lexer errors with line and column information with help to specify which part of the compiler the error message is coming from.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 lexer_error() [1/3]

```
ylang::lexer_error::lexer_error (
            const std::string & msg )  [inline]
```

Parameters

msg	the error message

Note

This constructor is used for debugging purposes only and helps determine if the program is behaving as expected

7.27.2.2 lexer_error() [2/3]

```
ylang::lexer_error::lexer_error (
            uint32_t line,
            uint32_t col,
            const std::string & msg )  [inline]
```

Parameters

line	the line number of the error
col	the column number of the error
msg	the error message

Note

This constructor is used for debugging purposes only and helps determine if the program is behaving as expected

7.27.2.3 lexer_error() [3/3]

```
ylang::lexer_error::lexer_error (
            ErrorLevel level,
            uint32_t line,
            uint32_t col,
            const std::string & msg )  [inline]
```

Parameters

level	the level of the error
line	the line number of the error
col	the column number of the error
msg	the error message

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

- YLang0.2/include/errors.hpp

7.28 ylang::Parser Class Reference

Public Member Functions

- **Parser** (std::shared_ptr< ErrorHandler > err_handler, std::vector< Token > lexemes)
- std::vector< std::shared_ptr< ASTStmnt > > **Parse** ()
- const bool **ParseFailed** () const

Private Member Functions

- parser_error **GetError** (ErrorLevel level, const std::string &msg)
- Token **Peek** () const
- Token **Previous** () const
- Token **Advance** ()
- Token **Consume** (Type t, const std::string &msg)
- bool **IsAtEnd** () const
- bool **Check** (Type type)
- bool **Match** (std::vector< Type > types)
- void **Synchronize** ()
- std::shared_ptr< ASTLiteral > **CreateTypedLiteral** (Token token)
- std::shared_ptr< ASTStmnt > **ParseDecl** ()
- std::shared_ptr< ASTStmnt > **ParseVarDecl** ()
- std::shared_ptr< ASTStmnt > **ParseStmnt** ()
- std::shared_ptr< ASTStmnt > **ParsePrintStmnt** ()
- std::shared_ptr< ASTStmnt > **ParseExprStmnt** ()
- std::shared_ptr< ASTStmnt > **ParseBlockStmnt** ()
- std::shared_ptr< ASTStmnt > **ParseIfStmnt** ()
- std::shared_ptr< ASTStmnt > **ParseWhileStmnt** ()
- std::shared_ptr< ASTExpr > **ParseExpr** ()
- std::shared_ptr< ASTExpr > **ParseAssignment** ()
- std::shared_ptr< ASTExpr > **ParseLogicalOr** ()
- std::shared_ptr< ASTExpr > **ParseLogicalAnd** ()
- std::shared_ptr< ASTExpr > **ParseEquality** ()
- std::shared_ptr< ASTExpr > **ParseComparison** ()
- std::shared_ptr< ASTExpr > **ParseTerm** ()
- std::shared_ptr< ASTExpr > **ParseFactor** ()
- std::shared_ptr< ASTExpr > **ParseUnary** ()
- std::shared_ptr< ASTExpr > **ParsePrimary** ()

Private Attributes

- std::shared_ptr< ErrorHandler > **err_handler** = nullptr
- Type **var_type** = Type::ERROR_TOKEN
- bool **in_var_decl** = false
- uint32_t **index** = 0
- bool **abort_parse** = false
- std::vector< Token > **lexemes** {}
- std::vector< std::shared_ptr< ASTStmnt > > **statements** {}

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

- YLang0.2/include/parser.hpp
- YLang0.2/src/parser.cpp

7.29 ylang::parser_error Class Reference

Class used to throw parser errors with line and column information with help to specify which part of the compiler the error message is coming from.

`#include <errors.hpp>`

Inheritance diagram for ylang::parser_error:

Collaboration diagram for ylang::parser_error:

Public Member Functions

- parser_error (const std::string &msg)
- parser_error (uint32_t line, uint32_t col, const std::string &msg)
- parser_error (ErrorLevel level, uint32_t line, uint32_t col, const std::string &msg)

Public Attributes

- ErrorLevel **level** = ErrorLevel::UNDEFINED_BEHAVIOR
- uint32_t **line** = 0
- uint32_t **column** = 0

7.29.1 Detailed Description

Class used to throw parser errors with line and column information with help to specify which part of the compiler the error message is coming from.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 parser_error() [1/3]

```
ylang::parser_error::parser_error (
            const std::string & msg )  [inline]
```

Parameters

msg	the error message

Note

This constructor is used for debugging purposes only and helps determine if the program is behaving as expected

7.29.2.2 parser_error() [2/3]

```
ylang::parser_error::parser_error (
            uint32_t line,
            uint32_t col,
            const std::string & msg )  [inline]
```

Parameters

line	the line number of the error
col	the column number of the error
msg	the error message

Note

This constructor is used for debugging purposes only and helps determine if the program is behaving as expected

7.29.2.3 parser_error() [3/3]

```
ylang::parser_error::parser_error (
            ErrorLevel level,
            uint32_t line,
            uint32_t col,
            const std::string & msg )  [inline]
```

Parameters

level	the level of the error
line	the line number of the error
col	the column number of the error
msg	the error message

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

- YLang0.2/include/errors.hpp

7.30 ylang::Token Struct Reference

```
#include <defines.hpp>
```

Public Member Functions

- **Token** (Type type, uint32_t line, uint32_t column, std::string token)

Public Attributes

- Type **type**
- uint32_t **line**
- uint32_t **column**
- std::string **token**

7.30.1 Detailed Description

This is the token, which contains all the information about a token after it is converted from a RawToken object.

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

- YLang0.2/include/defines.hpp

7.31 ylang::Util Class Reference

used to read and validate the source code from a file as well as act as a store-all for miscallaneous utility functions

`#include <util.hpp>`

Static Public Member Functions

- static const std::unique_ptr< CmndLineArgs > & **GetCmndLineArgs** ()
- static const void PrintUsage ()

  See Y Lang documentation for more information.
- static const void **PrintVersion** ()
- static void **ReadCmndLineArgs** (int *argc, char *argv[ ])
- static void PrintToken (const Token &token)
- static std::string ReadSrc ()
- static const std::string GetCurrFileName ()

Static Public Attributes

- static std::unique_ptr< CmndLineArgs > **cmnd_line_args** = nullptr

7.31.1 Detailed Description

used to read and validate the source code from a file as well as act as a store-all for miscallaneous utility functions

7.31.2 Member Function Documentation

7.31.2.1 GetCurrFileName()

```
const std::string ylang::Util::GetCurrFileName ( )  [static]
```

Returns

name of the current file if the -f flag is present, "<no-file.yl>" otherwise

7.31.2.2 PrintToken()

```
void ylang::Util::PrintToken (
            const Token & token )  [static]
```

Parameters

token	token to print

7.31.2.3 PrintUsage()

```
const void ylang::Util::PrintUsage ( )  [static]
```

See Y Lang documentation for more information.

7.31.2.4 ReadSrc()

```
std::string ylang::Util::ReadSrc ( )  [static]
```

Parameters

file_path	path to the file to read the source code from

Returns

the source code from the file

Exceptions

runtime	error if the file does not exist

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

- YLang0.2/include/util.hpp
- YLang0.2/src/util.cpp

7.32 ylang::Variable Struct Reference

Public Attributes

- VarType **type**
- VarValue **value**

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

- YLang0.2/include/interpreter.hpp

7.33 ylang::VM Struct Reference

Static Public Member Functions

- static void **Interpret** (const std::vector< std::shared_ptr< ASTStmnt >> &ast)

Static Public Attributes

- static std::shared_ptr< ErrorHandler > **err_handler** = std::make_shared<ErrorHandler>()
- static std::stack< Variable > **vstack** {}
- static std::stack< std::unordered_map< std::string, Variable > > **variables** {}

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

- YLang0.2/include/interpreter.hpp
- YLang0.2/src/interpreter.cpp

Chapter 8

File Documentation

8.1 YLang0.2/include/defines.hpp File Reference

contains main definitions and other core information

```
#include <string>
#include <cassert>
#include <variant>
```

Include dependency graph for defines.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct ylang::Token

Namespaces

- ylang

  contains all the ylang compiler related code

Macros

- #define **DEBUG_PRINT**(x) (void)0;
- #define **UNIMPLIMENTED** (void)0;
- #define **UNIMPLIMENTED_DEBUG**(x) (void)0;
- #define **UNIMPLIMENTED_RETURN**(x) (void)0;
- #define **UNIMPLIMENTED_RETURN_DEBUG**(msg, x) (void)0;
- #define **ASSERT**(x) (void)0;
- #define **PRINT_TOKEN**(t) (void)0;

Enumerations

- enum class **Type** {
  START_OF_FILE , **KW_I8** , **KW_I16** , **KW_I32** ,
  KW_I64 , **KW_U8** , **KW_U16** , **KW_U32** ,
  KW_U64 , **KW_F32** , **KW_F64** , **KW_BOOL** ,
  KW_CHAR , **KW_STRING** , **KW_BYTE** , **KW_NULL** ,
  KW_VOID , **KW_FN** , **KW_IF** , **KW_ELSE** ,
  KW_FOR , **KW_WHILE** , **KW_SWITCH** , **KW_RETURN** ,
  KW_BREAK , **KW_CONTINUE** , **KW_TRUE** , **KW_FALSE** ,
  KW_STRUCT , **KW_CLASS** , **KW_OBJECT** , **KW_ENUM** ,
  KW_UNION , **KW_NAMESPACE** , **KW_USING** , **KW_IMPORT** ,
  KW_EXPORT , **KW_PUBLIC** , **KW_PRIVATE** , **KW_INCLUDE** ,
  KW_PRINT , **IDENTIFIER** , **OP_ADD** , **OP_SUB** ,
  OP_MUL , **OP_DIV** , **OP_MOD** , **OP_OPEN_PAREN** ,
  OP_CLOSE_PAREN , **OP_OPEN_BRACE** , **OP_CLOSE_BRACE** , **OP_OPEN_BRACKET** ,
  OP_CLOSE_BRACKET , **OP_SEMICOLON** , **OP_COMMA** , **OP_DOT** ,
  OP_COLON , **OP_DOLLAR** , **OP_HASH** , **OP_AT** ,
  OP_EXCLAMATION , **OP_QUESTION** , **OP_AMPERSAND** , **OP_ASSIGN** ,
  OP_LESS_THAN , **OP_GREATER_THAN** , **OP_BIT_OR** , **OP_BIT_AND** ,
  OP_XOR , **OP_BIT_NOT** , **OP_ADD_ASSIGN** , **OP_SUB_ASSIGN** ,
  OP_MUL_ASSIGN , **OP_DIV_ASSIGN** , **OP_MOD_ASSIGN** , **OP_EQUALS** ,
  OP_NOT_EQUAL , **OP_LESS_THAN_EQUAL** , **OP_GREATER_THAN_EQUAL** , **OP_LOGICAL_AND** ,
  OP_LOGICAL_OR , **OP_BIT_OR_ASSIGN** , **OP_BIT_AND_ASSIGN** , **OP_XOR_ASSIGN** ,
  OP_BIT_NOT_ASSIGN , **OP_LEFT_SHIFT** , **OP_RIGHT_SHIFT** , **OP_LEFT_SHIFT_ASSIGN** ,
  OP_RIGHT_SHIFT_ASSIGN , **OP_INCREMENT** , **OP_DECREMENT** , **OP_ARROW** ,
  OP_SCOPE , **OP_CAPTURE** , **LIT_INT** , **LIT_HEX** ,
  LIT_FLOAT , **LIT_CHAR** , **LIT_STRING** , **COMMENT** ,
  ERROR_TOKEN , **END_OF_FILE** }

Variables

- static const std::string **ylang::YLANG_VERSION** = "0.0.2"
- static const std::string **ylang::TokenTypeStrings** [ ]

8.1.1 Detailed Description

contains main definitions and other core information

Version

0.0.2

Date

7-7-2023

Author

Y

8.2 YLang0.2/include/errors.hpp File Reference

contains major error handling code and classes including the error handler class and different error classes

```
#include <string>
#include <stdexcept>
#include <stack>
#include <mutex>
```

Include dependency graph for errors.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ylang::lexer_error

  Class used to throw lexer errors with line and column information with help to specify which part of the compiler the error message is coming from.
- class ylang::parser_error

  Class used to throw parser errors with line and column information with help to specify which part of the compiler the error message is coming from.
- class ylang::interpreter_error

  Class used to throw compiler errors with line and column information with help to specify which part of the compiler the error message is coming from.
- class ylang::compiler_error

  Class used to throw compiler errors with line and column information with help to specify which part of the compiler the error message is coming from.
- struct ylang::Error

  Struct used to store errors in the error handler.
- class ylang::ErrorHandler

  Class used to handle errors.

Namespaces

- ylang

  contains all the ylang compiler related code

Enumerations

- enum class **ErrorLevel** { **WARNING** , **FATAL** , **UNDEFINED_BEHAVIOR** }

8.2.1 Detailed Description

contains major error handling code and classes including the error handler class and different error classes

Version

0.0.2

Date

7-7-2023

Author

Y

8.3 YLang0.2/include/lexer.hpp File Reference

Lexer class header file, defines the interface for the Lexer class used internally by the ylang compiler.

```
#include <string>
#include <vector>
#include <map>
#include "defines.hpp"
#include "errors.hpp"
```

Include dependency graph for lexer.hpp:

8.4 YLang0.2/include/util.hpp File Reference

contains miscallaneous utility functions

```
#include <string>
#include <map>
#include <optional>
#include <memory>
#include <vector>
#include <iostream>
#include "defines.hpp"
```

Include dependency graph for util.hpp:

Classes

- class ylang::CmndLineArgs

  used to parse the command line arguments
- class ylang::Util

  used to read and validate the source code from a file as well as act as a store-all for miscallaneous utility functions

Namespaces

- ylang

  contains all the ylang compiler related code

8.4.1 Detailed Description

contains miscallaneous utility functions

Version

0.0.2

Date

7-7-2023

Author

Y

Index

~CmndLineArgs
    ylang::CmndLineArgs, 28

Advance
    ylang::Lexer, 42
ArgHasValue
    ylang::CmndLineArgs, 28
ArgPresent
    ylang::CmndLineArgs, 28
args
    ylang::CmndLineArgs, 29

CmndLineArgs
    ylang::CmndLineArgs, 27
compiler_error
    ylang::compiler_error, 30, 31
Consume
    ylang::Lexer, 42

Error
    ylang::Error, 32–34

flags
    ylang::CmndLineArgs, 29
FlushErrors
    ylang::ErrorHandler, 36

GetArgValue
    ylang::CmndLineArgs, 28
GetCurrFileName
    ylang::Util, 50
GetTokens
    ylang::Lexer, 42

HandleComment
    ylang::Lexer, 43
HandleLexerError
    ylang::ErrorHandler, 36
HandleParserError
    ylang::ErrorHandler, 36
HandleRuntimeError
    ylang::ErrorHandler, 36
HandleString
    ylang::Lexer, 43

InitTokenTypeMaps
    ylang::Lexer, 43
interpreter_error
    ylang::interpreter_error, 39, 40
IsEmpty
    ylang::ErrorHandler, 36

keywords
    ylang::Lexer, 44

level
    ylang::Error, 34
Lex
    ylang::Lexer, 43
Lexer
    ylang::Lexer, 42
lexer_error
    ylang::lexer_error, 45, 46

NewLine
    ylang::Lexer, 44

operators
    ylang::Lexer, 44

parser_error
    ylang::parser_error, 48, 49
Peek
    ylang::Lexer, 44
PeekNext
    ylang::Lexer, 44
Print
    ylang::AST, 15
PrintToken
    ylang::Util, 51
PrintUsage
    ylang::Util, 51

ReadSrc
    ylang::Util, 51

SubmitError
    ylang::ErrorHandler, 36

VisitVarAssignExpr
    ylang::Interpreter, 38

ylang, 11
YLang0.2/include/defines.hpp, 53
YLang0.2/include/errors.hpp, 55
YLang0.2/include/lexer.hpp, 56
YLang0.2/include/util.hpp, 56
ylang::AST, 15
    Print, 15
ylang::ASTBinaryExpr, 16
ylang::ASTBlockStmnt, 16
ylang::ASTExpr, 17
ylang::ASTExprStmnt, 17

ylang::ASTGroupingExpr, 18
ylang::ASTIfStmnt, 19
ylang::ASTLiteral, 19
ylang::ASTLogicalExpr, 20
ylang::ASTNode, 21
ylang::ASTPrinter, 21
ylang::ASTPrintStmnt, 22
ylang::ASTStmnt, 22
ylang::ASTUnaryExpr, 23
ylang::ASTVarAssignExpr, 24
ylang::ASTVarExpr, 24
ylang::ASTVarStmnt, 25
ylang::ASTWhileStmnt, 26
ylang::CmndLineArgs, 26
    ~CmndLineArgs, 28
    ArgHasValue, 28
    ArgPresent, 28
    args, 29
    CmndLineArgs, 27
    flags, 29
    GetArgValue, 28
ylang::compiler_error, 30
    compiler_error, 30, 31
ylang::Error, 31
    Error, 32–34
    level, 34
ylang::ErrorHandler, 35
    FlushErrors, 36
    HandleLexerError, 36
    HandleParserError, 36
    HandleRuntimeError, 36
    IsEmpty, 36
    SubmitError, 36
ylang::ExprVisitor, 37
ylang::Interpreter, 37
    VisitVarAssignExpr, 38
ylang::interpreter_error, 39
    interpreter_error, 39, 40
ylang::Lexer, 40
    Advance, 42
    Consume, 42
    GetTokens, 42
    HandleComment, 43
    HandleString, 43
    InitTokenTypeMaps, 43
    keywords, 44
    Lex, 43
    Lexer, 42
    NewLine, 44
    operators, 44
    Peek, 44
    PeekNext, 44
ylang::lexer_error, 45
    lexer_error, 45, 46
ylang::Parser, 47
ylang::parser_error, 48
    parser_error, 48, 49
ylang::Token, 49
ylang::Util, 50
    GetCurrFileName, 50
    PrintToken, 51
    PrintUsage, 51
    ReadSrc, 51
ylang::Variable, 52
ylang::VM, 52