MathScript

Generated by Doxygen 1.9.6

1 ReadMe 1

2 Hierarchical Index 3

2.1 Class Hierarchy . . . 3

3 Class Index 5

3.1 Class List . . . 5

4 File Index 7

4.1 File List . . . 7

5 Class Documentation 9

5.1 AddExpr Class Reference . . . 9

5.1.1 Detailed Description . . . 9

5.1.2 Constructor & Destructor Documentation . . . 10

5.1.2.1 AddExpr() . . . 10

5.1.3 Member Function Documentation . . . 10

5.1.3.1 equals() . . . 10

5.1.3.2 pretty_print_at() . . . 10

5.1.3.3 print() . . . 11

5.2 BoolExpr Class Reference . . . 11

5.2.1 Detailed Description . . . 12

5.2.2 Constructor & Destructor Documentation . . . 12

5.2.2.1 BoolExpr() . . . 12

5.2.3 Member Function Documentation . . . 12

5.2.3.1 equals() . . . 12

5.2.3.2 pretty_print_at() . . . 13

5.2.3.3 print() . . . 13

5.3 BoolValue Class Reference . . . 13

5.3.1 Detailed Description . . . 14

5.3.2 Constructor & Destructor Documentation . . . 14

5.3.2.1 BoolValue() . . . 14

5.3.3 Member Function Documentation . . . 14

5.3.3.1 equals() . . . 14

5.3.3.2 is_true() . . . 16

5.3.3.3 print() . . . 16

5.4 CallExpr Class Reference . . . 16

5.4.1 Detailed Description . . . 17

5.4.2 Constructor & Destructor Documentation . . . 17

5.4.2.1 CallExpr() . . . 17

5.4.3 Member Function Documentation . . . 18

5.4.3.1 equals() . . . 18

5.4.3.2 pretty_print_at() . . . 18

5.4.3.3 print() . . . 18

5.5 EmptyEnv Class Reference . . . 19
5.5.1 Member Function Documentation . . . 19
5.5.1.1 PTR() . . . 19
5.6 Env Class Reference . . . 20
5.7 EqExpr Class Reference . . . 20
5.7.1 Detailed Description . . . 21
5.7.2 Constructor & Destructor Documentation . . . 21
5.7.2.1 EqExpr() . . . 21
5.7.3 Member Function Documentation . . . 21
5.7.3.1 equals() . . . 21
5.7.3.2 pretty_print_at() . . . 21
5.7.3.3 print() . . . 22
5.8 ExtendedEnv Class Reference . . . 22
5.8.1 Member Function Documentation . . . 23
5.8.1.1 PTR() [1/2] . . . 23
5.8.1.2 PTR() [2/2] . . . 23
5.9 FunExpr Class Reference . . . 23
5.9.1 Detailed Description . . . 24
5.9.2 Constructor & Destructor Documentation . . . 24
5.9.2.1 FunExpr() . . . 24
5.9.3 Member Function Documentation . . . 24
5.9.3.1 equals() . . . 25
5.9.3.2 pretty_print_at() . . . 25
5.9.3.3 print() . . . 25
5.10 FunValue Class Reference . . . 26
5.10.1 Detailed Description . . . 26
5.10.2 Constructor & Destructor Documentation . . . 27
5.10.2.1 FunValue() . . . 27
5.10.3 Member Function Documentation . . . 27
5.10.3.1 equals() . . . 27
5.10.3.2 is_true() . . . 27
5.10.3.3 print() . . . 28
5.11 IfExpr Class Reference . . . 28
5.11.1 Detailed Description . . . 29
5.11.2 Constructor & Destructor Documentation . . . 29
5.11.2.1 IfExpr() . . . 29
5.11.3 Member Function Documentation . . . 29
5.11.3.1 equals() . . . 29
5.11.3.2 pretty_print_at() . . . 30
5.11.3.3 print() . . . 30
5.12 LetExpr Class Reference . . . 30
5.12.1 Detailed Description . . . 31

5.12.2 Constructor & Destructor Documentation . . . 31
5.12.2.1 LetExpr() . . . 31
5.12.3 Member Function Documentation . . . 32
5.12.3.1 equals() . . . 32
5.12.3.2 pretty_print_at() . . . 32
5.12.3.3 print() . . . 32
5.13 mainWidget Class Reference . . . 33
5.14 MultExpr Class Reference . . . 34
5.14.1 Detailed Description . . . 34
5.14.2 Constructor & Destructor Documentation . . . 34
5.14.2.1 MultExpr() . . . 34
5.14.3 Member Function Documentation . . . 35
5.14.3.1 equals() . . . 35
5.14.3.2 pretty_print_at() . . . 35
5.14.3.3 print() . . . 36
5.15 NumExpr Class Reference . . . 36
5.15.1 Detailed Description . . . 37
5.15.2 Constructor & Destructor Documentation . . . 37
5.15.2.1 NumExpr() . . . 37
5.15.3 Member Function Documentation . . . 37
5.15.3.1 equals() . . . 37
5.15.3.2 pretty_print_at() . . . 37
5.15.3.3 print() . . . 38
5.16 NumValue Class Reference . . . 38
5.16.1 Detailed Description . . . 39
5.16.2 Constructor & Destructor Documentation . . . 39
5.16.2.1 NumValue() . . . 39
5.16.3 Member Function Documentation . . . 39
5.16.3.1 equals() . . . 39
5.16.3.2 is_true() . . . 41
5.16.3.3 print() . . . 41
5.17 VarExpr Class Reference . . . 41
5.17.1 Detailed Description . . . 42
5.17.2 Constructor & Destructor Documentation . . . 42
5.17.2.1 VarExpr() . . . 42
5.17.3 Member Function Documentation . . . 43
5.17.3.1 equals() . . . 43
5.17.3.2 pretty_print_at() . . . 43
5.17.3.3 print() . . . 43

6 File Documentation **45**

6.1 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Env.h . . . 45

6.2 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.cpp File Reference 45
6.2.1 Detailed Description 46
6.2.2 Function Documentation 46
6.2.2.1 PTR() 46
6.3 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h File Reference 48
6.3.1 Detailed Description 49
6.3.2 Enumeration Type Documentation 49
6.3.2.1 precedence_t 49
6.3.3 Function Documentation 50
6.3.3.1 CLASS() 50
6.4 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h 50
6.5 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/mainwidget.h 52
6.6 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Parse.cpp File Reference 53
6.6.1 Detailed Description 53
6.6.2 Function Documentation 54
6.6.2.1 parse_keyword() 54
6.6.2.2 PTR() [1/2] 54
6.6.2.3 PTR() [2/2] 56
6.6.2.4 skip_whitespace() 56
6.7 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Parse.h File Reference 56
6.7.1 Detailed Description 57
6.7.2 Function Documentation 57
6.7.2.1 parse_keyword() 57
6.7.2.2 PTR() [1/2] 57
6.7.2.3 PTR() [2/2] 59
6.7.2.4 skip_whitespace() 60
6.8 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Parse.h 60
6.9 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/pointer.h File Reference 60
6.9.1 Detailed Description 61
6.10 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/pointer.h 61
6.11 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Value.cpp File Reference 61
6.11.1 Detailed Description 62
6.11.2 Function Documentation 62
6.11.2.1 PTR() 62
6.12 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Value.h File Reference 64
6.12.1 Detailed Description 64
6.12.2 Function Documentation 64
6.12.2.1 CLASS() 65
6.13 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Value.h 65

Index **67**

Chapter 1

ReadMe

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Env 20
- EmptyEnv 19
- ExtendedEnv 22

Expr
- AddExpr 9
- BoolExpr 11
- CallExpr 16
- EqExpr 20
- FunExpr 23
- IfExpr 28
- LetExpr 30
- MultExpr 34
- NumExpr 36
- VarExpr 41

QWidget
- mainWidget 33

Value
- BoolValue 13
- FunValue 26
- NumValue 38

Chapter 3

Class Index

3.1 Class List

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

AddExpr
: AddExpr class that represents an addition expression, inherits from Expr . . . 9

BoolExpr
: BoolExpr class that represents boolean expressions . . . 11

BoolValue
: BoolValue class that represents a true/false value, inherits from Value . . . 13

CallExpr
: CallExpr class that represents a call to a function . . . 16

EmptyEnv . . . 19

Env . . . 20

EqExpr
: EqExpr class that represents equality between two Exprs . . . 20

ExtendedEnv . . . 22

FunExpr
: FunExpr class that represents a function . . . 23

FunValue
: FunValue class that represents a function value, inherits from Value . . . 26

IfExpr
: IfExpr class that represents an if expressions (e.g. if...then...else...) . . . 28

LetExpr
: LetExpr class that represents let phrases (e.g. let x = 5 in x + 5), inherits from Expr . . . 30

mainWidget . . . 33

MultExpr
: MultExpr class that represents a multiplication expression, inherits from Expr . . . 34

NumExpr
: NumExpr class that represents a number, inherits from Expr . . . 36

NumValue
: NumValue class that represents a number value, inherits from Value . . . 38

VarExpr
: VarExpr class that represents a variable, inherits from Expr . . . 41

Chapter 4

File Index

4.1 File List

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

/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Env.h . . . . . **??**
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.cpp
Contains the definitions of the Expr class along with its children 45
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h
Expression class 48
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/mainwidget.h . . . . . **??**
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Parse.cpp
Contains functions used to parse input from the user or a file 53
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Parse.h
Parse declaration 56
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/pointer.h
Header file containing macro definitions about which pointer system to use 60
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Value.cpp
Contains the definitions of the Value class along with its children 61
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Value.h
Value class 64

Chapter 5

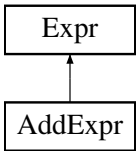
Class Documentation

5.1 AddExpr Class Reference

AddExpr class that represents an addition expression, inherits from Expr.

`#include <Expr.h>`

Inheritance diagram for AddExpr:



Public Member Functions

- **PTR** (Expr) lhs

  The Expr on the left-hand side of the AddExpr class.
- **PTR** (Expr) rhs

  The Expr on the right-hand side of the AddExpr class.
- AddExpr (PTR(Expr) lhs, PTR(Expr) rhs)

  Constructor.
- bool equals (PTR(Expr) e)

  Equals method checks to see if two expressions are equal.
- **PTR** (Value) interp(PTR(Env) env)
- void print (std::ostream &os)

  Print method that prints the AddExpr.
- void pretty_print_at (std::ostream &os, precedence_t precedence, bool needKeywordParenthesis, std↩ ::streampos &position)

  A method to print out the AddExpr in a more visually pleasing way.

5.1.1 Detailed Description

AddExpr class that represents an addition expression, inherits from Expr.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AddExpr()

```
AddExpr::AddExpr (
            PTR(Expr) lhs,
            PTR(Expr) rhs )
```

Constructor.

Parameters

Expr	lhs = Left hand side expression
Expr	rhs = right hand side expression

5.1.3 Member Function Documentation

5.1.3.1 equals()

```
bool AddExpr::equals (
            PTR(Expr) e )
```

Equals method checks to see if two expressions are equal.

Parameters

Expr	e = expression to be checked against

Returns

false if not equal or null, true if equal

5.1.3.2 pretty_print_at()

```
void AddExpr::pretty_print_at (
            std::ostream & os,
            precedence_t precedence,
            bool needKeywordParenthesis,
            std::streampos & position )
```

A method to print out the AddExpr in a more visually pleasing way.

Parameters

os	the stream to print out the expression
precedence	the precedence of the expression
needKeywordParenthesis	a boolean to indicate whether or not to include parenthesis in an expression with keywords
position	the current position of the stream that is printing out

5.1.3.3 print()

```
void AddExpr::print (
            std::ostream & os )
```

Print method that prints the AddExpr.

Parameters

ostream	&os: the ostream to print to

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

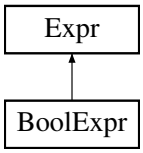
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.cpp

5.2 BoolExpr Class Reference

BoolExpr class that represents boolean expressions.

`#include <Expr.h>`

Inheritance diagram for BoolExpr:



Public Member Functions

- BoolExpr (bool val)

  Constructor for the BoolExpr class.
- bool equals (PTR(Expr) e)

  Equals method checks two expressions to see if they are equal.
- **PTR** (Value) interp(PTR(Env) env)
- void print (std::ostream &os)

  Print method that prints the BoolExpr.
- void pretty_print_at (std::ostream &os, precedence_t precedence, bool needKeywordParenthesis, std::streampos &position)

  A method to print out the BoolExpr expression in a more visually pleasing way.

Public Attributes

- bool **val**

  The boolean value of the BoolExpr class.

5.2.1 Detailed Description

BoolExpr class that represents boolean expressions.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 BoolExpr()

```
BoolExpr::BoolExpr (
            bool val )
```

Constructor for the BoolExpr class.

Parameters

val	the boolean value for the BoolExpr class

5.2.3 Member Function Documentation

5.2.3.1 equals()

```
bool BoolExpr::equals (
            PTR(Expr) e )
```

Equals method checks two expressions to see if they are equal.

Parameters

Expr	e = expression to be checked against

Returns

false if not equal or null, true if equal

5.2.3.2 pretty_print_at()

```
void BoolExpr::pretty_print_at (
            std::ostream & os,
            precedence_t precedence,
            bool needKeywordParenthesis,
            std::streampos & position )
```

A method to print out the BoolExpr expression in a more visually pleasing way.

Parameters

os	the stream to print out the expression
precedence	the precedence of the expression
needKeywordParenthesis	a boolean to indicate whether or not to include parenthesis in an expression with keywords
position	the current position of the stream that is printing out

5.2.3.3 print()

```
void BoolExpr::print (
            std::ostream & os )
```

Print method that prints the BoolExpr.

Parameters

ostream	&os: the ostream to print to

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

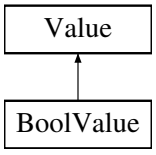
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.cpp

5.3 BoolValue Class Reference

BoolValue class that represents a true/false value, inherits from Value.

```
#include <Value.h>
```

Inheritance diagram for BoolValue:



Public Member Functions

- BoolValue (bool val)

  A constructor for the BoolValue class.
- bool equals (PTR(Value) otherValue)
- **PTR** (Value) add_to(PTR(Value) otherValue)
- **PTR** (Value) multiply_with(PTR(Value) otherValue)
- void print (std::ostream &os)

  A method to print out the value of a BoolValue.
- bool is_true ()

  A method to tell if a BoolValue is true or not.
- **PTR** (Value) call(PTR(Value) actual_arg)

Public Attributes

- bool **val**

  The boolean value of the BoolValue class.

5.3.1 Detailed Description

BoolValue class that represents a true/false value, inherits from Value.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 BoolValue()

```
BoolValue::BoolValue (
            bool val )
```

A constructor for the BoolValue class.

Parameters

val	the boolean used to indicate the BoolValue's boolean value

5.3.3 Member Function Documentation

5.3.3.1 equals()

```
bool BoolValue::equals (
            PTR(Value) otherValue )
```

\breif A method to test if two BoolValues are equal

Parameters

otherValue	the other Value to be tested against

Returns

true if they are equal, false if they are not or incompatible

5.3.3.2 is_true()

```
bool BoolValue::is_true ( )
```

A method to tell if a BoolValue is true or not.

Returns

the val of the BoolValue, indicating true or false

5.3.3.3 print()

```
void BoolValue::print (
            std::ostream & os )
```

A method to print out the value of a BoolValue.

Parameters

os	the output stream to print to

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

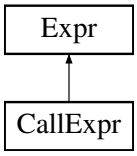
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Value.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Value.cpp

5.4 CallExpr Class Reference

CallExpr class that represents a call to a function.

```
#include <Expr.h>
```

Inheritance diagram for CallExpr:



Public Member Functions

- **PTR** (Expr) to_be_called

  The expression to be called.
- **PTR** (Expr) actual_arg

  The expression to be passed to the function.
- CallExpr (PTR(Expr) to_be_called, PTR(Expr) actual_arg)

  Constructor.
- bool equals (PTR(Expr) e)

  A method to test if two Expressions are equal.
- **PTR** (Value) interp(PTR(Env) env)
- void print (std::ostream &os)

  A method used to print out a call expression.
- void pretty_print_at (std::ostream &os, precedence_t precedence, bool needKeywordParenthesis, std::streampos &position)

  A method used to print out a call expression in a more visually pleasing way.

5.4.1 Detailed Description

CallExpr class that represents a call to a function.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 CallExpr()

```
CallExpr::CallExpr (
            PTR(Expr) to_be_called,
            PTR(Expr) actual_arg )
```

Constructor.

Parameters

var	the expression representing the function to be called
body	the expression representing the argument to be passed to the function

5.4.3 Member Function Documentation

5.4.3.1 equals()

```
bool CallExpr::equals (
            PTR(Expr) e )
```

A method to test if two Expressions are equal.

Parameters

e	the expression to be compared to

Returns

true if the expressions are equal, false if not

5.4.3.2 pretty_print_at()

```
void CallExpr::pretty_print_at (
            std::ostream & os,
            precedence_t precedence,
            bool needKeywordParenthesis,
            std::streampos & position )
```

A method used to print out a call expression in a more visually pleasing way.

Parameters

os	the output stream to be printed to
precedence	the precedence of the expression
needKeywordParenthesis	whether or not the keywords need parentheses
position	the position of the cursor

5.4.3.3 print()

```
void CallExpr::print (
            std::ostream & os )
```

A method used to print out a call expression.

Parameters

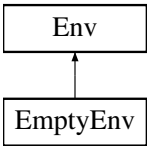
os	the output stream to be printed to

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

- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.cpp

5.5 EmptyEnv Class Reference

Inheritance diagram for EmptyEnv:



Public Member Functions

- PTR (Value) lookup(string find_name)
- virtual **PTR** (Value) lookup(string find_name)=0

Additional Inherited Members

Static Public Member Functions inherited from Env

- static **PTR** (Env) empty

5.5.1 Member Function Documentation

5.5.1.1 PTR()

```
EmptyEnv::PTR (
            Value  )  [inline], [virtual]
```

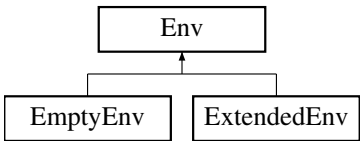
Implements Env.

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

- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Env.h

5.6 Env Class Reference

Inheritance diagram for Env:



Public Member Functions

- virtual **PTR** (Value) lookup(string find_name)=0

Static Public Member Functions

- static **PTR** (Env) empty

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

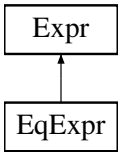
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Env.h

5.7 EqExpr Class Reference

EqExpr class that represents equality between two Exprs.

`#include <Expr.h>`

Inheritance diagram for EqExpr:



Public Member Functions

- **PTR** (Expr) lhs

  The expression on the left hand side of the equals.
- **PTR** (Expr) rhs

  The expression on the right hands side of the equals.
- EqExpr (PTR(Expr) lhs, PTR(Expr) rhs)

  Constructor.
- bool equals (PTR(Expr) e)

  Equals method checks two expressions to see if they are equal.
- **PTR** (Value) interp(PTR(Env) env)
- void print (std::ostream &os)

  Print method that prints the EqExpr.
- void pretty_print_at (std::ostream &os, precedence_t precedence, bool needKeywordParenthesis, std::streampos &position)

  A method to print out the EqExpr expression in a more visually pleasing way.

5.7.1 Detailed Description

EqExpr class that represents equality between two Exprs.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 EqExpr()

```
EqExpr::EqExpr (
            PTR(Expr) lhs,
            PTR(Expr) rhs )
```

Constructor.

Parameters

lhs	the expression on the left side of the equals
rhs	the expression on the right side of the equals

5.7.3 Member Function Documentation

5.7.3.1 equals()

```
bool EqExpr::equals (
            PTR(Expr) e )
```

Equals method checks two expressions to see if they are equal.

Parameters

Expr	e = expression to be checked against

Returns

false if not equal or null, true if equal

5.7.3.2 pretty_print_at()

```
void EqExpr::pretty_print_at (
            std::ostream & os,
```

```
            precedence_t precedence,
            bool needKeywordParenthesis,
            std::streampos & position )
```

A method to print out the EqExpr expression in a more visually pleasing way.

Parameters

os	the stream to print out the expression
precedence	the precedence of the expression
needKeywordParenthesis	a boolean to indicate whether or not to include parenthesis in an expression with keywords
position	the current position of the stream that is printing out

5.7.3.3 print()

```
void EqExpr::print (
            std::ostream & os )
```

Print method that prints the EqExpr.

Parameters

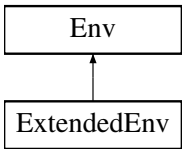
ostream	&os: the ostream to print to

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

- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.cpp

5.8 ExtendedEnv Class Reference

Inheritance diagram for ExtendedEnv:



Public Member Functions

- PTR (Value) val
- **PTR** (Env) rest
- **ExtendedEnv** (string name, PTR(Value) val, PTR(Env) rest)
- PTR (Value) lookup(string find_name)

- virtual **PTR** (Value) lookup(string find_name)=0

Public Attributes

- string **name**

Additional Inherited Members

Static Public Member Functions inherited from Env

- static **PTR** (Env) empty

5.8.1 Member Function Documentation

5.8.1.1 PTR() [1/2]

```
ExtendedEnv::PTR (
            Value  )  [inline], [virtual]
```

Implements Env.

5.8.1.2 PTR() [2/2]

```
ExtendedEnv::PTR (
            Value  )  [virtual]
```

Implements Env.

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

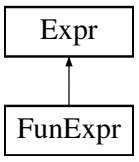
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Env.h

5.9 FunExpr Class Reference

FunExpr class that represents a function.

```
#include <Expr.h>
```

Inheritance diagram for FunExpr:



Public Member Functions

- **PTR** (Expr) body

  The function that will contain the variable that will be evaluated.
- FunExpr (std::string formal_arg, PTR(Expr) body)

  Constructor.
- bool equals (PTR(Expr) e)

  A method to check if two Expressions are equal.
- **PTR** (Value) interp(PTR(Env) env)
- void print (std::ostream &os)

  A method used to print out the function expression.
- void pretty_print_at (std::ostream &os, precedence_t precedence, bool needKeywordParenthesis, std↩ ::streampos &position)

  A method used to print out a function expression in a more visually pleasing way.

Public Attributes

- std::string **formal_arg**

  The variable that will be replaced in the function.

5.9.1 Detailed Description

FunExpr class that represents a function.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 FunExpr()

```
FunExpr::FunExpr (
            std::string formal_arg,
            PTR(Expr) body )
```

Constructor.

Parameters

formal_arg	the string that will represent the formal argument of the function
body	the expression that represents the function to be analyzed

5.9.3 Member Function Documentation

5.9.3.1 equals()

```
bool FunExpr::equals (
            PTR(Expr) e )
```

A method to check if two Expressions are equal.

Parameters

e	the expression to be checked against

Returns

true if the expressions are equal, false if not

5.9.3.2 pretty_print_at()

```
void FunExpr::pretty_print_at (
            std::ostream & os,
            precedence_t precedence,
            bool needKeywordParenthesis,
            std::streampos & position )
```

A method used to print out a function expression in a more visually pleasing way.

Parameters

os	the output stream to print to
precedence	the precedence of the expression
needKeywordParenthesis	whether or not keyword parentheses are needed
position	the position of the cursor

5.9.3.3 print()

```
void FunExpr::print (
            std::ostream & os )
```

A method used to print out the function expression.

Parameters

os	the output stream to print to

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

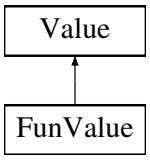
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.cpp

5.10 FunValue Class Reference

FunValue class that represents a function value, inherits from Value.

`#include <Value.h>`

Inheritance diagram for FunValue:



Public Member Functions

- **PTR** (Expr) body

  The expression that the formal argument will replace to interpret.
- **PTR** (Env) env

  The environmne to pass along.
- FunValue (std::string formal_arg, PTR(Expr) body, PTR(Env) env)

  Constructor.
- **PTR** (Expr) to_expr()
- bool equals (PTR(Value) otherValue)

  A method to evaluate equality between two Values.
- **PTR** (Value) add_to(PTR(Value) otherValue)
- **PTR** (Value) multiply_with(PTR(Value) otherValue)
- void print (std::ostream &os)

  A method to print a FunValue.
- bool is_true ()

  A method to evaluate if the FunValue is true.
- **PTR** (Value) call(PTR(Value) actual_arg)

Public Attributes

- std::string **formal_arg**

  The formal argument of the function.

5.10.1 Detailed Description

FunValue class that represents a function value, inherits from Value.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 FunValue()

```
FunValue::FunValue (
            std::string formal_arg,
            PTR(Expr) body,
            PTR(Env) env )
```

Constructor.

Parameters

formal_arg	the formal argument of the function
body	the function to be analyzed

5.10.3 Member Function Documentation

5.10.3.1 equals()

```
bool FunValue::equals (
            PTR(Value) otherValue )
```

A method to evaluate equality between two Values.

Parameters

otherValue	the other value to be compared to this value

Returns

true if the values are equal

5.10.3.2 is_true()

```
bool FunValue::is_true ( )
```

A method to evaluate if the FunValue is true.

Returns

an error, since a FunValue cannot be a boolean

5.10.3.3 print()

```
void FunValue::print (
            std::ostream & os )
```

A method to print a FunValue.

Parameters

os	the stream to print out to

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

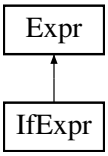
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Value.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Value.cpp

5.11 IfExpr Class Reference

IfExpr class that represents an if expressions (e.g. if...then...else...)

```
#include <Expr.h>
```

Inheritance diagram for IfExpr:



Public Member Functions

- **PTR** (Expr) condition

  The condition used in the IfExpr.
- **PTR** (Expr) rhs

  The expression representing the "then" statement.
- **PTR** (Expr) lhs

  The expression representing the "else" statement.
- IfExpr (PTR(Expr) condition, PTR(Expr) rhs, PTR(Expr) lhs)

  Constructor.
- bool equals (PTR(Expr) e)

  Equals method checks two expressions to see if they are equal.
- **PTR** (Value) interp(PTR(Env) env)
- void print (std::ostream &os)

  Print method that prints the IfExpr.
- void pretty_print_at (std::ostream &os, precedence_t precedence, bool needKeywordParenthesis, std::streampos &position)

  A method to print out the IfExpr expression in a more visually pleasing way.

5.11.1 Detailed Description

IfExpr class that represents an if expressions (e.g. if...then...else...)

5.11.2 Constructor & Destructor Documentation

5.11.2.1 IfExpr()

```
IfExpr::IfExpr (
            PTR(Expr) condition,
            PTR(Expr) rhs,
            PTR(Expr) lhs )
```

Constructor.

Parameters

condition	the expression used as the condition in the IfExpr
rhs	the expression used as the "then" statement
lhs	the expression used as the "else" statement

5.11.3 Member Function Documentation

5.11.3.1 equals()

```
bool IfExpr::equals (
            PTR(Expr) e )
```

Equals method checks two expressions to see if they are equal.

Parameters

Expr	e = expression to be checked against

Returns

false if not equal or null, true if equal

5.11.3.2 pretty_print_at()

```
void IfExpr::pretty_print_at (
            std::ostream & os,
            precedence_t precedence,
            bool needKeywordParenthesis,
            std::streampos & position )
```

A method to print out the IfExpr expression in a more visually pleasing way.

Parameters

os	the stream to print out the expression
precedence	the precedence of the expression
needKeywordParenthesis	a boolean to indicate whether or not to include parenthesis in an expression with keywords
position	the current position of the stream that is printing out

5.11.3.3 print()

```
void IfExpr::print (
            std::ostream & os )
```

Print method that prints the IfExpr.

Parameters

ostream	&os: the ostream to print to

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

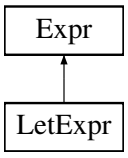
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.cpp

5.12 LetExpr Class Reference

LetExpr class that represents let phrases (e.g. let x = 5 in x + 5), inherits from Expr.

`#include <Expr.h>`

Inheritance diagram for LetExpr:



Public Member Functions

- **PTR** (Expr) rhs

  The "left-hand side" of the let (the thing that will replace val)
- **PTR** (Expr) body

  The "right-hand side" of the let (the "in" part, denoting what contains val)
- LetExpr (std::string val, PTR(Expr) rhs, PTR(Expr) body)

  Constructor.
- bool equals (PTR(Expr) e)

  Equals method checks two LetExpr expressions to see if they are equal.
- **PTR** (Value) interp(PTR(Env) env)
- void print (std::ostream &os)

  Print method that prints the LetExpr.
- void pretty_print_at (std::ostream &os, precedence_t precedence, bool needKeywordParenthesis, std↩ ::streampos &position)

  A method to print out the LetExpr expression in a more visually pleasing way.

Public Attributes

- std::string **val**

  The variable expression that will be replaced with an Expr.

5.12.1 Detailed Description

LetExpr class that represents let phrases (e.g. let x = 5 in x + 5), inherits from Expr.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 LetExpr()

```
LetExpr::LetExpr (
            std::string val,
            PTR(Expr) rhs,
            PTR(Expr) body )
```

Constructor.

Parameters

val	the string value that explains what will be replaced
rhs	the expression that will replace val
rhs	the expression that contains val that will be replaced

5.12.3 Member Function Documentation

5.12.3.1 equals()

```
bool LetExpr::equals (
            PTR(Expr) e )
```

Equals method checks two LetExpr expressions to see if they are equal.

Parameters

Expr	e = expression to be checked against

Returns

false if not equal or null, true if equal

5.12.3.2 pretty_print_at()

```
void LetExpr::pretty_print_at (
            std::ostream & os,
            precedence_t precedence,
            bool needKeywordParenthesis,
            std::streampos & position )
```

A method to print out the LetExpr expression in a more visually pleasing way.

Parameters

os	the stream to print out the expression
precedence	the precedence of the expression
needKeywordParenthesis	a boolean to indicate whether or not to include parenthesis in an expression with keywords
position	the current position of the stream that is printing out

5.12.3.3 print()

```
void LetExpr::print (
            std::ostream & os )
```

Print method that prints the LetExpr.

Parameters

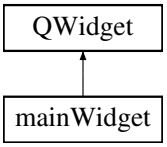
ostream	&os: the ostream to print to

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

- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.cpp

5.13 mainWidget Class Reference

Inheritance diagram for mainWidget:



Public Slots

- void **submitPush** ()
- void **resetPush** ()

Public Member Functions

- **mainWidget** (QWidget ∗parent=nullptr)

Public Attributes

- QVBoxLayout ∗ **vertical1**
- QHBoxLayout ∗ **horizontal1**
- QLabel ∗ **expression**
- QVBoxLayout ∗ **vertical2**
- QTextEdit ∗ **input**
- QGroupBox ∗ **groupBox**
- QGridLayout ∗ **grid**
- QRadioButton ∗ **interpButton**
- QRadioButton ∗ **prettyPrintButton**
- QLabel ∗ **interp**
- QLabel ∗ **prettyPrint**
- QPushButton ∗ **submit**
- QHBoxLayout ∗ **horizontal2**
- QLabel ∗ **result**
- QVBoxLayout ∗ **vertical3**
- QTextEdit ∗ **output**
- QPushButton ∗ **reset**

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

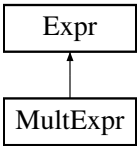
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/mainwidget.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/mainwidget.cpp

5.14 MultExpr Class Reference

MultExpr class that represents a multiplication expression, inherits from Expr.

`#include <Expr.h>`

Inheritance diagram for MultExpr:



Public Member Functions

- **PTR** (Expr) lhs

  The Expr on the left-hand side of the MultExpr class.
- **PTR** (Expr) rhs

  The Expr on the right-hand side of the MultExpr class.
- MultExpr (PTR(Expr) lhs, PTR(Expr) rhs)

  Constructor.
- bool equals (PTR(Expr) e)

  Equals method checks two expressions to see if they are equal.
- **PTR** (Value) interp(PTR(Env) env)
- void print (std::ostream &os)

  Print method that prints the MultExpr.
- void pretty_print_at (std::ostream &os, precedence_t precedence, bool needKeywordParenthesis, std::streampos &position)

  A method to print out the MultExpr in a more visually pleasing way.

5.14.1 Detailed Description

MultExpr class that represents a multiplication expression, inherits from Expr.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 MultExpr()

```
MultExpr::MultExpr (
            PTR(Expr) lhs,
            PTR(Expr) rhs )
```

Constructor.

Parameters

Expr	lhs = left hand side expression
Expr	rhs = right hand side expression

5.14.3 Member Function Documentation

5.14.3.1 equals()

```
bool MultExpr::equals (
            PTR(Expr) e )
```

Equals method checks two expressions to see if they are equal.

Parameters

Expr	e = expression to be checked against

Returns

false if not equal or null, true if equal

5.14.3.2 pretty_print_at()

```
void MultExpr::pretty_print_at (
            std::ostream & os,
            precedence_t precedence,
            bool needKeywordParenthesis,
            std::streampos & position )
```

A method to print out the MultExpr in a more visually pleasing way.

Parameters

os	the stream to print out the expression
precedence	the precedence of the expression
needKeywordParenthesis	a boolean to indicate whether or not to include parenthesis in an expression with keywords
position	the current position of the stream that is printing out

5.14.3.3 print()

```
void MultExpr::print (
            std::ostream & os )
```

Print method that prints the MultExpr.

Parameters

ostream	&os: the ostream to print to

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

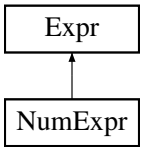
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.cpp

5.15 NumExpr Class Reference

NumExpr class that represents a number, inherits from Expr.

```
#include <Expr.h>
```

Inheritance diagram for NumExpr:



Public Member Functions

- NumExpr (int val)

  Constructor.
- bool equals (PTR(Expr) e)

  Equals method checks to see if a NumExpr is equal to another expression.
- **PTR** (Value) interp(PTR(Env) env)
- void print (std::ostream &os)

  Print method that prints the NumExpr.
- void pretty_print_at (std::ostream &os, precedence_t precedence, bool needKeywordParenthesis, std::streampos &position)

  A method to print out the NumExpr in a more visually pleasing way.

Public Attributes

- int **val**

  The integer value of the NumExpr class.

5.15.1 Detailed Description

NumExpr class that represents a number, inherits from Expr.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 NumExpr()

```
NumExpr::NumExpr (
            int val )
```

Constructor.

Parameters

int	val: the integer value assigned to the NumExpr

5.15.3 Member Function Documentation

5.15.3.1 equals()

```
bool NumExpr::equals (
            PTR(Expr) e )
```

Equals method checks to see if a NumExpr is equal to another expression.

Parameters

Expr	e: the Expression to compare to the NumExpr

Returns

false if not equal or null, true if equal

5.15.3.2 pretty_print_at()

```
void NumExpr::pretty_print_at (
            std::ostream & os,
```

```
            precedence_t precedence,
            bool needKeywordParenthesis,
            std::streampos & position )
```

A method to print out the NumExpr in a more visually pleasing way.

Parameters

os	the stream to print out the expression
precedence	the precedence of the expression
needKeywordParenthesis	a boolean to indicate whether or not to include parenthesis in an expression with keywords
position	the current position of the stream that is printing out

5.15.3.3 print()

```
void NumExpr::print (
            std::ostream & os )
```

Print method that prints the NumExpr.

Parameters

ostream	os: the ostream to print to

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

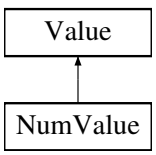
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.cpp

5.16 NumValue Class Reference

NumValue class that represents a number value, inherits from Value.

```
#include <Value.h>
```

Inheritance diagram for NumValue:



Public Member Functions

- NumValue (int val)

  Constructor.
- bool equals (PTR(Value) otherValue)
- **PTR** (Value) add_to(PTR(Value) otherValue)
- **PTR** (Value) multiply_with(PTR(Value) otherValue)
- void print (std::ostream &os)

  A method to print out the value of the NumValue as a string.
- bool is_true ()

  A method used to determine if the NumValue is true.
- **PTR** (Value) call(PTR(Value) actual_arg)

Public Attributes

- int **val**

  The integer value of the NumValue class.

5.16.1 Detailed Description

NumValue class that represents a number value, inherits from Value.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 NumValue()

```
NumValue::NumValue (
            int val )
```

Constructor.

Parameters

int	val: the integer value assigned to the NumValue

5.16.3 Member Function Documentation

5.16.3.1 equals()

```
bool NumValue::equals (
            PTR(Value) otherValue )
```

\breif A method to test if two NumValues are equal

Parameters

otherValue	the other Value to be tested against

Returns

true if they are equal, false if they are not or incompatible

5.16.3.2 is_true()

```
bool NumValue::is_true ( )
```

A method used to determine if the NumValue is true.

Returns

an error, since it is impossible for an int to be a boolean

5.16.3.3 print()

```
void NumValue::print (
            std::ostream & os )
```

A method to print out the value of the NumValue as a string.

Parameters

os	the stream to print out to

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

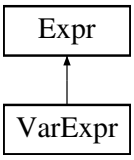
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Value.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Value.cpp

5.17 VarExpr Class Reference

VarExpr class that represents a variable, inherits from Expr.

```
#include <Expr.h>
```

Inheritance diagram for VarExpr:



Public Member Functions

- VarExpr (std::string val)

  Constructor.
- bool equals (PTR(Expr) e)

  Equals method checks two expressions to see if they are equal.
- **PTR** (Value) interp(PTR(Env) env)
- void print (std::ostream &os)

  Print method that prints the VarExpr.
- void pretty_print_at (std::ostream &os, precedence_t precedence, bool needKeywordParenthesis, std::streampos &position)

  A method to print out the VarExpr expression in a more visually pleasing way.

Public Attributes

- std::string **val**

  The string value of the VarExpr class.

5.17.1 Detailed Description

VarExpr class that represents a variable, inherits from Expr.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 VarExpr()

```
VarExpr::VarExpr (
            std::string val )
```

Constructor.

Parameters

String	val = string used to denote the value of the variable

5.17.3 Member Function Documentation

5.17.3.1 equals()

```
bool VarExpr::equals (
            PTR(Expr) e )
```

Equals method checks two expressions to see if they are equal.

Parameters

Expr	e = expression to be checked against

Returns

false if not equal or null, true if equal

5.17.3.2 pretty_print_at()

```
void VarExpr::pretty_print_at (
            std::ostream & os,
            precedence_t precedence,
            bool needKeywordParenthesis,
            std::streampos & position )
```

A method to print out the VarExpr expression in a more visually pleasing way.

Parameters

os	the stream to print out the expression
precedence	the precedence of the expression
needKeywordParenthesis	a boolean to indicate whether or not to include parenthesis in an expression with keywords
position	the current position of the stream that is printing out

5.17.3.3 print()

```
void VarExpr::print (
            std::ostream & os )
```

Print method that prints the VarExpr.

Parameters

ostream	&os: the ostream to print to

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

- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h
- /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.cpp

Chapter 6

File Documentation

6.1 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Env.h

```
00001 //
00002 // Created by Levi Neely on 4/11/23.
00003 //
00004
00005 #ifndef EXPR_CPP_ENV_H
00006 #define EXPR_CPP_ENV_H
00007
00008 #include "pointer.h"
00009 #include <string>
00010 class Value;
00011
00012 using namespace std;
00013
00014 class Env {
00015 public:
00016     virtual PTR(Value) lookup(string find_name) = 0;
00017     static PTR(Env) empty;
00018 };
00019
00020 class EmptyEnv : public Env {
00021 public:
00022     PTR(Value) lookup(string find_name) {
00023         throw runtime_error("Free variable: " + find_name);
00024     }
00025 };
00026
00027 class ExtendedEnv : public Env {
00028 public:
00029     string name;
00030     PTR(Value) val;
00031     PTR(Env) rest;
00032     ExtendedEnv(string name, PTR(Value) val, PTR(Env) rest) {
00033         this->name = name;
00034         this->val = val;
00035         this->rest = rest;
00036     };
00037     PTR(Value) lookup(string find_name) {
00038         if (find_name == name) {
00039             return val;
00040         }
00041         else {
00042             return rest->lookup(find_name);
00043         }
00044     }
00045 };
00046
00047 #endif //EXPR_CPP_ENV_H
```

6.2 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/↩ Expr.cpp File Reference

Contains the definitions of the Expr class along with its children.

```
#include "Value.h"
#include "Expr.h"
#include <string>
#include <stdexcept>
#include <sstream>
```

Functions

- PTR (Value) NumExpr

  Interpretation method that computes the value of the expression.

6.2.1 Detailed Description

Contains the definitions of the Expr class along with its children.

This file contains the definitions of the Expr class and its children (NumExpr, AddExpr, MultExpr, VarExpr, and LetExpr) along with all of their various methods.

Author

Levi Neely

6.2.2 Function Documentation

6.2.2.1 PTR()

```
PTR (
            Value  )
```

Interpretation method that computes the value of the expression.

A method to multiply two values together.

A method to add two values together.

A method used to multiply two BoolValues together.

A method used to add two BoolValues together.

A method used to evaluate a function.

A method to multiply one NumValue with another.

A method to interpret the value of the CallExpr.

A method to interpret the FunExpr.

Method to interpret an EqExpr.

Method to interpret an IfExpr.

Interpretation method that returns the value of the expression.

Interpretation method that returns the value of the add expression.

Returns

the value of the expression

the value of the two expressions that make up the AddExpr

the value of the product of this expression

a Value, but since this is a variable, it will throw an error

a Value

a BoolValue, since it is a BoolExpr

the value of the IfExpr

true if both sides are interpreted as equal, false if not

a FunValue representing the FunExpr

a value representing the value of the CallExpr

Parameters

otherValue	the other NumValue to be multiplied with the original

Returns

the multiplied NumValue

Parameters

actual_arg	the argument to be implemented in the function

Returns

an error, since it is impossible to call a NumValue

Parameters

otherValue	the other value to add to this one

Returns

an error, since it is impossible to add booleans together

Parameters

otherValue	the other value to multiply with this one

Returns

an error, since it is impossible to multiply booleans together

Parameters

actual_arg	the argument to be implemented in the function

Returns

an error, since it is impossible to call a BoolValue

Parameters

otherValue	the other value to be added

Returns

an error, since it will not be able to add two FunValues together

Parameters

otherValue	the other value to be multiplied

Returns

an error, since it will not be able to multiply two FunValues together

Parameters

actual_arg	the argument to be implemented in the function

Returns

a value representing the function utilizing the actual argument

6.3 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h File Reference

Expression class.

```
#include "pointer.h"
#include "Env.h"
#include <string>
#include <sstream>
#include <iostream>
```

Classes

- class NumExpr

  NumExpr class that represents a number, inherits from Expr.
- class AddExpr

  AddExpr class that represents an addition expression, inherits from Expr.
- class MultExpr

  MultExpr class that represents a multiplication expression, inherits from Expr.
- class VarExpr

  VarExpr class that represents a variable, inherits from Expr.
- class LetExpr

  LetExpr class that represents let phrases (e.g. let x = 5 in x + 5), inherits from Expr.
- class BoolExpr

  BoolExpr class that represents boolean expressions.
- class IfExpr

  IfExpr class that represents an if expressions (e.g. if...then...else...)
- class EqExpr

  EqExpr class that represents equality between two Exprs.
- class FunExpr

  FunExpr class that represents a function.
- class CallExpr

  CallExpr class that represents a call to a function.

Enumerations

- enum precedence_t { prec_none , prec_eq , prec_add , prec_mult }

  enum to assign precedence for printing functions

Functions

- CLASS (Expr)

  Expr class that represents an expression.

6.3.1 Detailed Description

Expression class.

This file contains the declarations of the Expr class, its children, and all methods.

6.3.2 Enumeration Type Documentation

6.3.2.1 precedence_t

```
enum precedence_t
```

enum to assign precedence for printing functions

Enumerator

prec_none	No precedence.
prec_eq	Precedence for EqExpr.
prec_add	Precedence for AddExpr.
prec_mult	Precedence for MultExpr.

6.3.3 Function Documentation

6.3.3.1 CLASS()

```
CLASS (
            Expr  )
```

Expr class that represents an expression.

A method for all Expr that represents the Expr as a string

Returns

the string representing the Expr

A method that prints out Expr in a more visually pleasing way

Parameters

os	the output stream the Expr will be printed to

A method utilized for testing to return a string from the pretty_print method

Returns

a string representing an Expr printed using pretty_print

6.4 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h

Go to the documentation of this file.

```
00001
00008 #ifndef HW1_EXPR_H
00009 #define HW1_EXPR_H
00010 #include "pointer.h"
00011 #include "Env.h"
00012 #include <string>
00013 #include <sstream>
00014 #include <iostream>
00015 class Value;
00016
00020 typedef enum {
00021     prec_none,
```

```
00022     prec_eq,
00023     prec_add,
00024     prec_mult
00025 } precedence_t;
00026
00030 CLASS(Expr) {
00031 public:
00032     virtual bool equals(PTR(Expr) e) = 0;
00033     virtual PTR(Value) interp(PTR(Env) env) = 0;
00034     virtual void print(std::ostream& os) = 0;
00035
00040     std::string to_string() {
00041         std::stringstream st("");
00042         THIS->print(st);
00043         return st.str();
00044     };
00045
00050     void pretty_print(std::ostream& os) {
00051         precedence_t precedence = prec_none;
00052         std::streampos position = os.tellp();
00053         THIS->pretty_print_at(os, precedence, false, position);
00054     }
00055
00060     std::string pretty_print_to_string() {
00061         std::stringstream st("");
00062         pretty_print(st);
00063         return st.str();
00064     }
00065     virtual void pretty_print_at(std::ostream& os, precedence_t precedence, bool
      needKeywordParenthesis, std::streampos& position) = 0;
00066     virtual ~Expr() {};
00067 };
00068
00072 class NumExpr : public Expr {
00073 public:
00074     int val;
00075     NumExpr(int val);
00076     bool equals(PTR(Expr) e);
00077     PTR(Value) interp(PTR(Env) env);
00078     void print(std::ostream& os);
00079     void pretty_print_at(std::ostream& os, precedence_t precedence, bool needKeywordParenthesis,
      std::streampos& position);
00080 };
00081
00085 class AddExpr : public Expr {
00086 public:
00087     PTR(Expr) lhs;
00088     PTR(Expr) rhs;
00089     AddExpr(PTR(Expr) lhs, PTR(Expr) rhs);
00090     bool equals(PTR(Expr) e);
00091     PTR(Value) interp(PTR(Env) env);
00092     void print(std::ostream& os);
00093     void pretty_print_at(std::ostream& os, precedence_t precedence, bool needKeywordParenthesis,
      std::streampos& position);
00094 };
00095
00099 class MultExpr : public Expr {
00100 public:
00101     PTR(Expr) lhs;
00102     PTR(Expr) rhs;
00103     MultExpr(PTR(Expr) lhs, PTR(Expr) rhs);
00104     bool equals(PTR(Expr) e);
00105     PTR(Value) interp(PTR(Env) env);
00106     void print(std::ostream& os);
00107     void pretty_print_at(std::ostream& os, precedence_t precedence, bool needKeywordParenthesis,
      std::streampos& position);
00108 };
00109
00113 class VarExpr : public Expr {
00114 public:
00115     std::string val;
00116     VarExpr(std::string val);
00117     bool equals(PTR(Expr) e);
00118     PTR(Value) interp(PTR(Env) env);
00119     void print(std::ostream& os);
00120     void pretty_print_at(std::ostream& os, precedence_t precedence, bool needKeywordParenthesis,
      std::streampos& position);
00121 };
00122
00126  class LetExpr : public Expr {
00127  public:
00128      std::string val;
00129      PTR(Expr) rhs;
00130      PTR(Expr) body;
00131      LetExpr(std::string val, PTR(Expr) rhs, PTR(Expr) body);
00132      bool equals(PTR(Expr) e);
00133      PTR(Value) interp(PTR(Env) env);
```

```
00134      void print(std::ostream& os);
00135      void pretty_print_at(std::ostream& os, precedence_t precedence, bool needKeywordParenthesis,
      std::streampos& position);
00136  };
00137
00141  class BoolExpr : public Expr {
00142  public:
00143      bool val;
00144      BoolExpr(bool val);
00145      bool equals(PTR(Expr) e);
00146      PTR(Value) interp(PTR(Env) env);
00147      void print(std::ostream& os);
00148      void pretty_print_at(std::ostream& os, precedence_t precedence, bool needKeywordParenthesis,
      std::streampos& position);
00149  };
00150
00154  class IfExpr : public Expr {
00155  public:
00156      PTR(Expr) condition;
00157      PTR(Expr) rhs;
00158      PTR(Expr) lhs;
00159      IfExpr(PTR(Expr) condition, PTR(Expr) rhs, PTR(Expr) lhs);
00160      bool equals(PTR(Expr) e);
00161      PTR(Value) interp(PTR(Env) env);
00162      void print(std::ostream& os);
00163      void pretty_print_at(std::ostream& os, precedence_t precedence, bool needKeywordParenthesis,
      std::streampos& position);
00164  };
00165
00169  class EqExpr : public Expr {
00170  public:
00171      PTR(Expr) lhs;
00172      PTR(Expr) rhs;
00173      EqExpr(PTR(Expr) lhs, PTR(Expr) rhs);
00174      bool equals(PTR(Expr) e);
00175      PTR(Value) interp(PTR(Env) env);
00176      void print(std::ostream& os);
00177      void pretty_print_at(std::ostream& os, precedence_t precedence, bool needKeywordParenthesis,
      std::streampos& position);
00178  };
00179
00183  class FunExpr : public Expr {
00184  public:
00185     std::string formal_arg;
00186     PTR(Expr) body;
00187     FunExpr(std::string formal_arg, PTR(Expr) body);
00188     bool equals(PTR(Expr) e);
00189     PTR(Value) interp(PTR(Env) env);
00190     void print(std::ostream& os);
00191     void pretty_print_at(std::ostream& os, precedence_t precedence, bool needKeywordParenthesis,
      std::streampos& position);
00192  };
00193
00197  class CallExpr : public Expr {
00198  public:
00199      PTR(Expr) to_be_called;
00200      PTR(Expr) actual_arg;
00201      CallExpr(PTR(Expr) to_be_called, PTR(Expr) actual_arg);
00202      bool equals(PTR(Expr) e);
00203      PTR(Value) interp(PTR(Env) env);
00204      void print(std::ostream& os);
00205      void pretty_print_at(std::ostream& os, precedence_t precedence, bool needKeywordParenthesis,
      std::streampos& position);
00206  };
00207 #endif //HW1_EXPR_H
```

6.5 /Users/levineely/Portfolio/Portfolio/MathematicalScripting↩ App/mainwidget.h

```
00001 #ifndef MAINWIDGET_H
00002 #define MAINWIDGET_H
00003
00004 #include <QWidget>
00005 #include <QLabel>
00006 #include <QLineEdit>
00007 #include <QSpinBox>
00008 #include <QRadioButton>
00009 #include <QPushButton>
00010 #include <QTextEdit>
00011 #include <QVBoxLayout>
00012 #include <QGridLayout>
00013 #include <QHBoxLayout>
```

```
00014 #include <QGroupBox>
00015
00016 class mainWidget : public QWidget
00017 {
00018     Q_OBJECT
00019 public:
00020     explicit mainWidget(QWidget *parent = nullptr);
00021     QVBoxLayout *vertical1;
00022     QHBoxLayout *horizontal1;
00023     QLabel *expression;
00024     QVBoxLayout *vertical2;
00025     QTextEdit *input;
00026     QGroupBox *groupBox;
00027     QGridLayout *grid;
00028     QRadioButton *interpButton;
00029     QRadioButton *prettyPrintButton;
00030     QLabel *interp;
00031     QLabel *prettyPrint;
00032     QPushButton *submit;
00033     QHBoxLayout *horizontal2;
00034     QLabel *result;
00035     QVBoxLayout *vertical3;
00036     QTextEdit *output;
00037     QPushButton *reset;
00038 signals:
00039 public slots:
00040     void submitPush();
00041     void resetPush();
00042 };
00043
00044 #endif // MAINWIDGET_H
```

6.6 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Parse.cpp File Reference

Contains functions used to parse input from the user or a file.

```
#include "Parse.h"
#include "Expr.h"
#include <iostream>
```

Functions

- PTR (Expr) parse_expr(istream &in)

  Method used to parse an expression.
- string parse_keyword (istream &in)

  A helper method to parse individual keywords into strings.
- PTR (Expr) parse_string(string input)

  A helper method used to parse a string (mostly used for testing)
- void skip_whitespace (istream &in)

  A helper method used to skip any whitespace within the input to be parsed.

6.6.1 Detailed Description

Contains functions used to parse input from the user or a file.

This file contains the definitions of the methods used to parse either the input from a user or an input file in order to interpret, print, or pretty print an expression

Author

Levi Neely

6.6.2 Function Documentation

6.6.2.1 parse_keyword()

```
string parse_keyword (
            istream & in )
```

A helper method to parse individual keywords into strings.

Parameters

in	the source of the input

Returns

a string representing the keyword

6.6.2.2 PTR() [1/2]

```
PTR (
            Expr  ) &
```

Method used to parse an expression.

A method used to parse an IfExpr from an input stream.

A method used to parse a LetExpr expression.

A method used to parse a VarExpr expression.

A method used to parse a NumExpr expression.

A method used to parse an inner (another part of recursive parsing)

A method used to parse a multicand (another part of the recursive parsing)

A method used to parse an addend (another part of recursive parsing)

Method used to parse a comparg (a way to recursively parse input)

Parameters

in	the source of the input

Returns

a fully-parsed expression

Parameters

in	the source of the input

Returns

a parsed addend

Parameters

in	the source of the input

Returns

a parsed multicand

Parameters

in	the source of the input

Returns

a parsed inner

Parameters

in	the source of the input

Returns

a fully parsed NumExpr expression

Parameters

in	the source of the input

Returns

a fully-parsed VarExpr expression

Parameters

in	the source of the input

Returns

a fully-parsed LetExpr expression

Parameters

in	the input stream to read from

Returns

a fully parsed IfExpr

6.6.2.3 PTR() [2/2]

```
PTR (
            Expr  )
```

A helper method used to parse a string (mostly used for testing)

Parameters

input	the string used as the input to be parsed

Returns

a fully-parsed expression

6.6.2.4 skip_whitespace()

```
void skip_whitespace (
            istream & in )
```

A helper method used to skip any whitespace within the input to be parsed.

Parameters

in	the source of the input

6.7 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/↩ Parse.h File Reference

Parse declaration.

```
#include "pointer.h"
#include "Expr.h"
```

Functions

- void skip_whitespace (istream &in)

  A helper method used to skip any whitespace within the input to be parsed.
- PTR (Expr) parse_num(istream &in)

  Method used to parse an expression.
- PTR (Expr) parse_string(string input)

  A helper method used to parse a string (mostly used for testing)
- string parse_keyword (istream &in)

  A helper method to parse individual keywords into strings.

6.7.1 Detailed Description

Parse declaration.

This file contains the declaration of the various methods used in the Parsing Process.

6.7.2 Function Documentation

6.7.2.1 parse_keyword()

```
string parse_keyword (
            istream & in )
```

A helper method to parse individual keywords into strings.

Parameters

in	the source of the input

Returns

a string representing the keyword

6.7.2.2 PTR() [1/2]

```
PTR (
            Expr  ) &
```

Method used to parse an expression.

A method used to parse an IfExpr from an input stream.

A method used to parse a LetExpr expression.

A method used to parse a VarExpr expression.

A method used to parse a NumExpr expression.

A method used to parse an inner (another part of recursive parsing)

A method used to parse a multicand (another part of the recursive parsing)

A method used to parse an addend (another part of recursive parsing)

Method used to parse a comparg (a way to recursively parse input)

Parameters

in	the source of the input

Returns

a fully-parsed expression

Parameters

in	the source of the input

Returns

a parsed addend

Parameters

in	the source of the input

Returns

a parsed multicand

Parameters

in	the source of the input

Returns

a parsed inner

Parameters

in	the source of the input

Returns

a fully parsed NumExpr expression

Parameters

in	the source of the input

Returns

a fully-parsed VarExpr expression

Parameters

in	the source of the input

Returns

a fully-parsed LetExpr expression

Parameters

in	the input stream to read from

Returns

a fully parsed IfExpr

6.7.2.3 PTR() [2/2]

```
PTR (
            Expr  )
```

A helper method used to parse a string (mostly used for testing)

Parameters

input	the string used as the input to be parsed

Returns

a fully-parsed expression

6.7.2.4 skip_whitespace()

```
void skip_whitespace (
             istream & in )
```

A helper method used to skip any whitespace within the input to be parsed.

Parameters

in	the source of the input

6.8 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/↩ Parse.h

Go to the documentation of this file.

```
00001
00008 #ifndef EXPR_CPP_PARSE_H
00009 #define EXPR_CPP_PARSE_H
00010
00011 #include "pointer.h"
00012 #include "Expr.h"
00013
00014 using namespace std;
00015
00016 static void consume(istream &in, int expect);
00017 void skip_whitespace(istream &in);
00018 PTR(Expr) parse_num(istream &in);
00019 PTR(Expr) parse_comparg(istream &in);
00020 PTR(Expr) parse_addend(istream &in);
00021 PTR(Expr) parse_multicand(istream &in);
00022 PTR(Expr) parse_inner(istream &in);
00023 PTR(Expr) parse_expr(istream &in);
00024 PTR(Expr) parse_var(istream &in);
00025 PTR(Expr) parse_let(istream &in);
00026 PTR(Expr) parse_if(istream &in);
00027 PTR(Expr) parse_function(istream &in);
00028 PTR(Expr) parse_string(string input);
00029 string parse_keyword(istream &in);
00030
00031 #endif //EXPR_CPP_PARSE_H
```

6.9 /Users/levineely/Portfolio/Portfolio/MathematicalScripting↩ App/pointer.h File Reference

header file containing macro definitions about which pointer system to use

```
#include <memory>
```

Macros

- #define **USE_PLAIN_POINTERS** 0
- #define **NEW**(T) std::make_shared<T>
- #define **PTR**(T) std::shared_ptr<T>
- #define **CAST**(T) std::dynamic_pointer_cast<T>
- #define **CLASS**(T) class T : public std::enable_shared_from_this<T>
- #define **THIS** shared_from_this()

6.9.1 Detailed Description

header file containing macro definitions about which pointer system to use

This file contains the definition of macros used to select which pointer system to utilize inside the program (either smart pointers or classic pointers).

Author

Levi Neely

6.10 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/pointer.h

Go to the documentation of this file.

```
00001
00011 #ifndef EXPR_CPP_POINTER_H
00012 #define EXPR_CPP_POINTER_H
00013
00014 #include <memory>
00015
00016 #define USE_PLAIN_POINTERS 0
00017 #if USE_PLAIN_POINTERS
00018
00019 # define NEW(T)     new T
00020 # define PTR(T)     T*
00021 # define CAST(T)    dynamic_cast<T*>
00022 # define CLASS(T)   class T
00023 # define THIS       this
00024
00025 #else
00026
00027 # define NEW(T)     std::make_shared<T>
00028 # define PTR(T)     std::shared_ptr<T>
00029 # define CAST(T)    std::dynamic_pointer_cast<T>
00030 # define CLASS(T)   class T : public std::enable_shared_from_this<T>
00031 # define THIS       shared_from_this()
00032
00033 #endif
00034
00035 #endif //EXPR_CPP_POINTER_H
```

6.11 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Value.cpp File Reference

Contains the definitions of the Value class along with its children.

```
#include "Expr.h"
#include "Value.h"
#include <string>
```

Functions

- PTR (Value) NumValue

  A method to add one NumValue to another.

6.11.1 Detailed Description

Contains the definitions of the Value class along with its children.

This file contains the definitions of the Value class and its children (NumValue and BoolVal) along with all of their various methods.

Author

Levi Neely

6.11.2 Function Documentation

6.11.2.1 PTR()

```
PTR (
            Value  )
```

A method to add one NumValue to another.

A method to multiply two values together.

A method to add two values together.

A method used to multiply two BoolValues together.

A method used to add two BoolValues together.

A method used to evaluate a function.

A method to multiply one NumValue with another.

Parameters

otherValue	the other NumValue to be added to the original

Returns

the combined NumValue

Parameters

otherValue	the other NumValue to be multiplied with the original

Returns

the multiplied NumValue

Parameters

actual_arg	the argument to be implemented in the function

Returns

an error, since it is impossible to call a NumValue

Parameters

otherValue	the other value to add to this one

Returns

an error, since it is impossible to add booleans together

Parameters

otherValue	the other value to multiply with this one

Returns

an error, since it is impossible to multiply booleans together

Parameters

actual_arg	the argument to be implemented in the function

Returns

an error, since it is impossible to call a BoolValue

Parameters

otherValue	the other value to be added

Returns

an error, since it will not be able to add two FunValues together

Parameters

otherValue	the other value to be multiplied

Returns

an error, since it will not be able to multiply two FunValues together

Parameters

actual_arg	the argument to be implemented in the function

Returns

a value representing the function utilizing the actual argument

6.12 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/↩ Value.h File Reference

Value class.

```
#include "pointer.h"
#include "Env.h"
#include <string>
#include <sstream>
#include <iostream>
```

Classes

- class NumValue

  NumValue class that represents a number value, inherits from Value.
- class BoolValue

  BoolValue class that represents a true/false value, inherits from Value.
- class FunValue

  FunValue class that represents a function value, inherits from Value.

Functions

- CLASS (Value)

  Value class that represents a value.

6.12.1 Detailed Description

Value class.

This file contains the declarations of the Value class, its children, and all methods.

6.12.2 Function Documentation

6.12.2.1 CLASS()

```
CLASS (
            Value  )
```

Value class that represents a value.

A method for all Expr that represents the Expr as a string

Returns

the string representing the Expr

6.13 /Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/↩ Value.h

Go to the documentation of this file.

```
00001
00008 #ifndef EXPR_CPP_VALUE_H
00009 #define EXPR_CPP_VALUE_H
00010 #include "pointer.h"
00011 #include "Env.h"
00012 #include <string>
00013 #include <sstream>
00014 #include <iostream>
00015
00016 class Expr;
00020 CLASS(Value) {
00021 public:
00022     virtual bool equals(PTR(Value) otherValue) = 0;
00023     virtual PTR(Value) add_to(PTR(Value) otherValue) = 0;
00024     virtual PTR(Value) multiply_with(PTR(Value) otherValue) = 0;
00025     virtual void print(std::ostream& os) = 0;
00026     virtual bool is_true() = 0;
00027     virtual PTR(Value) call(PTR(Value) actual_arg) = 0;
00032     std::string to_string() {
00033         std::stringstream st("");
00034         THIS->print(st);
00035         return st.str();
00036     };
00037     virtual ~Value() {};
00038 };
00039
00043 class NumValue : public Value {
00044 public:
00045     int val;
00046     NumValue(int val);
00047     bool equals(PTR(Value) otherValue);
00048     PTR(Value) add_to(PTR(Value) otherValue);
00049     PTR(Value) multiply_with(PTR(Value) otherValue);
00050     void print(std::ostream& os);
00051     bool is_true();
00052     PTR(Value) call(PTR(Value) actual_arg);
00053 };
00054
00058 class BoolValue : public Value {
00059 public:
00060     bool val;
00061     BoolValue(bool val);
00062     bool equals(PTR(Value) otherValue);
00063     PTR(Value) add_to(PTR(Value) otherValue);
00064     PTR(Value) multiply_with(PTR(Value) otherValue);
00065     void print(std::ostream& os);
00066     bool is_true();
00067     PTR(Value) call(PTR(Value) actual_arg);
00068 };
00069
00073 class FunValue : public Value {
00074 public:
00075     std::string formal_arg;
00076     PTR(Expr) body;
00077     PTR(Env) env;
00078     FunValue(std::string formal_arg, PTR(Expr) body, PTR(Env) env);
00079     PTR(Expr) to_expr();
```

```
00080     bool equals(PTR(Value) otherValue);
00081     PTR(Value) add_to(PTR(Value) otherValue);
00082     PTR(Value) multiply_with(PTR(Value) otherValue);
00083     void print(std::ostream& os);
00084     bool is_true();
00085     PTR(Value) call(PTR(Value) actual_arg);
00086 };
00087 #endif //EXPR_CPP_VALUE_H
```

Index

/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.cpp, 45
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Expr.h, 48
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/ParseExpr.h, 53
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/ParseEnv.h, 56
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/ValueExpr.h, 61
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/Value.h, 64
/Users/levineely/Portfolio/Portfolio/MathematicalScriptingApp/powerExpr.h, 60

AddExpr, 9
  AddExpr, 10
  equals, 10
  pretty_print_at, 10
  print, 11

BoolExpr, 11
  BoolExpr, 12
  equals, 12
  pretty_print_at, 12
  print, 13
BoolValue, 13
  BoolValue, 14
  equals, 14
  is_true, 16
  print, 16

CallExpr, 16
  CallExpr, 17
  equals, 18
  pretty_print_at, 18
  print, 18
CLASS
  Expr.h, 50
  Value.h, 64

EmptyEnv, 19
  PTR, 19
Env, 20
EqExpr, 20
  EqExpr, 21
  equals, 21
  pretty_print_at, 21
  print, 22
equals
  AddExpr, 10
  BoolExpr, 12
  BoolValue, 14
  CallExpr, 18
  EqExpr, 21
  FunExpr, 24
  FunValue, 27
  IfExpr, 29
  LetExpr, 32
  MultExpr, 35
  NumExpr, 37
  NumValue, 39
  VarExpr, 43
Expr.cpp
  PTR, 46
Expr.h
  CLASS, 50
  prec_add, 50
  prec_eq, 50
  prec_mult, 50
  prec_none, 50
  precedence_t, 49
ExtendedEnv, 22
  PTR, 23

FunExpr, 23
  equals, 24
  FunExpr, 24
  pretty_print_at, 25
  print, 25
FunValue, 26
  equals, 27
  FunValue, 27
  is_true, 27
  print, 27

IfExpr, 28
  equals, 29
  IfExpr, 29
  pretty_print_at, 29
  print, 30
is_true
  BoolValue, 16
  FunValue, 27
  NumValue, 41

LetExpr, 30
  equals, 32
  LetExpr, 31
  pretty_print_at, 32

print, 32

mainWidget, 33
MultExpr, 34
equals, 35
MultExpr, 34
pretty_print_at, 35
print, 35

NumExpr, 36
equals, 37
NumExpr, 37
pretty_print_at, 37
print, 38
NumValue, 38
equals, 39
is_true, 41
NumValue, 39
print, 41

Parse.cpp
parse_keyword, 54
PTR, 54, 56
skip_whitespace, 56
Parse.h
parse_keyword, 57
PTR, 57, 59
skip_whitespace, 60
parse_keyword
Parse.cpp, 54
Parse.h, 57
prec_add
Expr.h, 50
prec_eq
Expr.h, 50
prec_mult
Expr.h, 50
prec_none
Expr.h, 50
precedence_t
Expr.h, 49
pretty_print_at
AddExpr, 10
BoolExpr, 12
CallExpr, 18
EqExpr, 21
FunExpr, 25
IfExpr, 29
LetExpr, 32
MultExpr, 35
NumExpr, 37
VarExpr, 43
print
AddExpr, 11
BoolExpr, 13
BoolValue, 16
CallExpr, 18
EqExpr, 22
FunExpr, 25
FunValue, 27
IfExpr, 30
LetExpr, 32
MultExpr, 35
NumExpr, 38
NumValue, 41
VarExpr, 43
PTR
EmptyEnv, 19
Expr.cpp, 46
ExtendedEnv, 23
Parse.cpp, 54, 56
Parse.h, 57, 59
Value.cpp, 62

skip_whitespace
Parse.cpp, 56
Parse.h, 60

Value.cpp
PTR, 62
Value.h
CLASS, 64
VarExpr, 41
equals, 43
pretty_print_at, 43
print, 43
VarExpr, 42