Msdscript

Generated by Doxygen 1.10.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

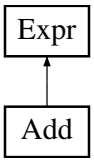
Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Add Class Reference

Inheritance diagram for Add:



Public Member Functions

- Add (Expr *lhs, Expr *rhs)

  Constructs an Add expression with the specified left and right expressions.
- bool equals (Expr *e) override

  Checks if the given expression equals this Add expression.
- int interp () const override

  Evaluates the Add expression and returns its value.
- bool has_variable () const override

  Checks if the Add expression has a variable.
- Expr * subst (const std::string &var, Expr *replacement) const override

  Substitutes a variable in the Add expression with a replacement expression.
- Expr * clone () const override

  Clones the Add expression.
- void print (std::ostream &os) const override

  Prints the Add expression to the output stream.
- void pretty_print (std::ostream &os) const override

  Pretty prints the Add expression.
- std::string to_pretty_string () const override

  Returns a pretty string representation of the Add expression.

Public Member Functions inherited from Expr

- std::string to_string () const

  Returns a string representation of the expression.
- virtual void pretty_print_at (std::ostream &os, precedence_t prec_current) const

  Pretty printsthe expression with varying parentheses.

Public Attributes

- Expr * **lhs**

  Represents left hand side.
- Expr * **rhs**

  Represents right hand side.

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Add()

```
Add::Add (
            Expr * lhs,
            Expr * rhs )
```

Constructs an Add expression with the specified left and right expressions.

Parameters

lhs	The left expression.
rhs	The right expression.

4.1.2 Member Function Documentation

4.1.2.1 clone()

```
Expr * Add::clone ( ) const  [override], [virtual]
```

Clones the Add expression.

Returns

A new copy of the Add expression.

Implements Expr.

4.1.2.2 equals()

```
bool Add::equals (
            Expr * e )  [override], [virtual]
```

Checks if the given expression equals this Add expression.

Parameters

e	The expression to compare with.

Returns

True if the expressions are equal, false otherwise.

Implements Expr.

4.1.2.3 has_variable()

```
bool Add::has_variable ( ) const  [override], [virtual]
```

Checks if the Add expression has a variable.

Returns

True if the Add expression has a variable, false otherwise.

Implements Expr.

4.1.2.4 interp()

```
int Add::interp ( ) const  [override], [virtual]
```

Evaluates the Add expression and returns its value.

Returns

The value of the Add expression.

Implements Expr.

4.1.2.5 pretty_print()

```
void Add::pretty_print (
            std::ostream & os ) const  [override], [virtual]
```

Pretty prints the Add expression.

Parameters

os	The output stream to print to.

Reimplemented from Expr.

4.1.2.6 print()

```
void Add::print (
            std::ostream & os ) const  [override], [virtual]
```

Prints the Add expression to the output stream.

Parameters

os	The output stream to print to.

Reimplemented from Expr.

4.1.2.7 subst()

```
Expr * Add::subst (
            const std::string & var,
            Expr * replacement ) const  [override], [virtual]
```

Substitutes a variable in the Add expression with a replacement expression.

Parameters

var	The variable to substitute.
replacement	The expression to replace the variable with.

Returns

A new expression after substitution.

Implements Expr.

4.1.2.8 to_pretty_string()

```
std::string Add::to_pretty_string ( ) const  [override], [virtual]
```

Returns a pretty string representation of the Add expression.

Returns

A string representing the Add expression.

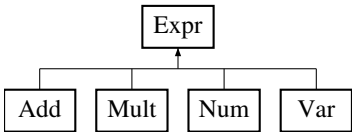
Reimplemented from Expr.

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

- expr.h
- expr.cpp

4.2 Expr Class Reference

Inheritance diagram for Expr:



Public Member Functions

- virtual bool equals (Expr *e)=0
- virtual int interp () const =0
- virtual bool has_variable () const =0
- virtual Expr * subst (const std::string &var, Expr *replacement) const =0
- virtual Expr * clone () const =0
- std::string to_string () const

  Returns a string representation of the expression.
- virtual void print (std::ostream &os) const

  Prints the expression to the output stream.
- virtual void pretty_print_at (std::ostream &os, precedence_t prec_current) const

  Pretty printsthe expression with varying parentheses.
- virtual void pretty_print (std::ostream &os) const

  calls recursive method pretty_print_at
- virtual std::string to_pretty_string () const

  Returns a pretty string representation of the expression.

4.2.1 Member Function Documentation

4.2.1.1 clone()

```
virtual Expr * Expr::clone ( ) const  [pure virtual]
```

Implemented in Num, Add, Mult, and Var.

4.2.1.2 equals()

```
virtual bool Expr::equals (
            Expr * e )  [pure virtual]
```

Implemented in Num, Add, Mult, and Var.

4.2.1.3 has_variable()

```
virtual bool Expr::has_variable ( ) const  [pure virtual]
```

Implemented in Num, Add, Mult, and Var.

4.2.1.4 interp()

```
virtual int Expr::interp ( ) const  [pure virtual]
```

Implemented in Num, Add, Mult, and Var.

4.2.1.5 pretty_print()

```
void Expr::pretty_print (
            std::ostream & os ) const  [virtual]
```

calls recursive method pretty_print_at

Parameters

os	The output stream to print to.

Reimplemented in Add, and Mult.

4.2.1.6 pretty_print_at()

```
void Expr::pretty_print_at (
            std::ostream & os,
            precedence_t prec_parent ) const  [virtual]
```

Pretty printsthe expression with varying parentheses.

Parameters

os	The output stream to print to.
prec_parent	The precedence of the parent expression.

4.2.1.7 print()

```
void Expr::print (
            std::ostream & os ) const  [virtual]
```

Prints the expression to the output stream.

Parameters

os	The output stream to print to.

Reimplemented in Num, Add, Mult, and Var.

4.2.1.8 subst()

```
virtual Expr * Expr::subst (
            const std::string & var,
            Expr * replacement ) const  [pure virtual]
```

Implemented in Num, Add, Mult, and Var.

4.2.1.9 to_pretty_string()

```
std::string Expr::to_pretty_string ( ) const  [virtual]
```

Returns a pretty string representation of the expression.

Parameters

none	

Returns

A string representing the expression.

Reimplemented in Add, and Mult.

4.2.1.10 to_string()

```
std::string Expr::to_string ( ) const
```

Returns a string representation of the expression.

Parameters

none	

Returns

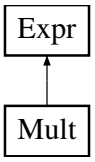
A string representing the expression.

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

- expr.h
- expr.cpp

4.3 Mult Class Reference

Inheritance diagram for Mult:



Public Member Functions

- Mult (Expr *lhs, Expr *rhs)

  Constructs a Mult expression with the specified left and right expressions.
- bool equals (Expr *e) override

  Checks if the given expression equals this Mult expression.
- int interp () const override

  Evaluates the Mult expression and returns its value.
- bool has_variable () const override

  Checks if the Mult expression has a variable.
- Expr * subst (const std::string &var, Expr *replacement) const override

  Substitutes a variable in the Mult expression with a replacement expression.
- Expr * clone () const override

  Clones the Mult expression.
- void print (std::ostream &os) const override

  Prints the Mult expression to the output stream.
- void pretty_print (std::ostream &os) const override

  Pretty prints the Mult expression.
- std::string to_pretty_string () const override

  Returns a pretty string representation of the Mult expression.

Public Member Functions inherited from Expr

- std::string to_string () const

  Returns a string representation of the expression.
- virtual void pretty_print_at (std::ostream &os, precedence_t prec_current) const

  Pretty printsthe expression with varying parentheses.

Public Attributes

- Expr * **lhs**

  Represents left hand side.
- Expr * **rhs**

  Represents right hand side.

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Mult()

```
Mult::Mult (
            Expr * lhs,
            Expr * rhs )
```

Constructs a Mult expression with the specified left and right expressions.

Parameters

lhs	The left expression.
rhs	The right expression.

4.3.2 Member Function Documentation

4.3.2.1 clone()

```
Expr * Mult::clone ( ) const  [override], [virtual]
```

Clones the Mult expression.

Returns

A new copy of the Mult expression.

Implements Expr.

4.3.2.2 equals()

```
bool Mult::equals (
            Expr * e )  [override], [virtual]
```

Checks if the given expression equals this Mult expression.

Parameters

e	The expression to compare with.

Returns

True if the expressions are equal, false otherwise.

Implements Expr.

4.3.2.3 has_variable()

```
bool Mult::has_variable ( ) const  [override], [virtual]
```

Checks if the Mult expression has a variable.

Returns

True if the Mult expression has a variable, false otherwise.

Implements Expr.

4.3.2.4 interp()

```
int Mult::interp ( ) const  [override], [virtual]
```

Evaluates the Mult expression and returns its value.

Returns

The value of the Mult expression.

Implements Expr.

4.3.2.5 pretty_print()

```
void Mult::pretty_print (
            std::ostream & os ) const  [override], [virtual]
```

Pretty prints the Mult expression.

Parameters

os	The output stream to print to.

Reimplemented from Expr.

4.3.2.6 print()

```
void Mult::print (
            std::ostream & os ) const  [override], [virtual]
```

Prints the Mult expression to the output stream.

Parameters

os	The output stream to print to.

Reimplemented from Expr.

4.3.2.7 subst()

```
Expr * Mult::subst (
            const std::string & var,
            Expr * replacement ) const  [override], [virtual]
```

Substitutes a variable in the Mult expression with a replacement expression.

Parameters

var	The variable to substitute.
replacement	The expression to replace the variable with.

Returns

A new expression after substitution.

Implements Expr.

4.3.2.8 to_pretty_string()

```
std::string Mult::to_pretty_string ( ) const  [override], [virtual]
```

Returns a pretty string representation of the Mult expression.

Returns

A string representing the Mult expression.

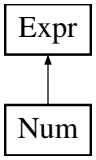
Reimplemented from Expr.

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

- expr.h
- expr.cpp

4.4 Num Class Reference

Inheritance diagram for Num:



Public Member Functions

- Num (int val)

  Constructs a Num object with the specified value.
- bool equals (Expr ∗e) override

  Checks if the given expression equals this Num object.
- int interp () const override

  Evaluates the Num expression and returns its value.
- bool has_variable () const override

  Checks if the Num expression has a variable.
- Expr ∗ subst (const std::string &var, Expr ∗replacement) const override

  Substitutes a variable in the Num expression with a replacement expression.
- Expr ∗ clone () const override

  Clones the Num expression.
- std::string to_string () const

  Returns a string representation of the Num expression.
- void print (std::ostream &os) const override

  Prints the Num expression to the output stream.

Public Member Functions inherited from Expr

- std::string to_string () const

  Returns a string representation of the expression.
- virtual void pretty_print_at (std::ostream &os, precedence_t prec_current) const

  Pretty printsthe expression with varying parentheses.
- virtual void pretty_print (std::ostream &os) const

  calls recursive method pretty_print_at
- virtual std::string to_pretty_string () const

  Returns a pretty string representation of the expression.

Public Attributes

- int **val**

  represents a number

4.4.1 Constructor & Destructor Documentation

4.4.1.1 Num()

```
Num::Num (
            int val )
```

Constructs a Num object with the specified value.

Parameters

val	The value of the Num object.

4.4.2 Member Function Documentation

4.4.2.1 clone()

```
Expr * Num::clone ( ) const  [override], [virtual]
```

Clones the Num expression.

Returns

A new copy of the Num expression.

Implements Expr.

4.4.2.2 equals()

```
bool Num::equals (
            Expr * e )  [override], [virtual]
```

Checks if the given expression equals this Num object.

Parameters

e	The expression to compare with.

Returns

True if the expressions are equal, false otherwise.

Implements Expr.

4.4.2.3 has_variable()

```
bool Num::has_variable ( ) const  [override], [virtual]
```

Checks if the Num expression has a variable.

Parameters

none	

Returns

True if the Num expression has a variable, false otherwise.

Implements Expr.

4.4.2.4 interp()

```
int Num::interp ( ) const  [override], [virtual]
```

Evaluates the Num expression and returns its value.

Parameters

none	

Returns

The value of the Num expression.

Implements Expr.

4.4.2.5 print()

```
void Num::print (
            std::ostream & os ) const  [override], [virtual]
```

Prints the Num expression to the output stream.

Parameters

os	The output stream to print to.

Reimplemented from Expr.

4.4.2.6 subst()

```
Expr * Num::subst (
            const std::string & var,
            Expr * replacement ) const  [override], [virtual]
```

Substitutes a variable in the Num expression with a replacement expression.

Parameters

var	The variable to substitute.
replacement	The expression to replace the variable with.

Returns

A new expression after substitution.

Implements Expr.

4.4.2.7 to_string()

```
std::string Num::to_string ( ) const
```

Returns a string representation of the Num expression.

Returns

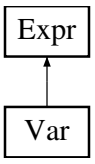
A string representing the Num expression.

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

- expr.h
- expr.cpp

4.5 Var Class Reference

Inheritance diagram for Var:



Public Member Functions

- Var (const std::string &varName)

  Constructs a Var expression with the specified variable name.
- bool equals (Expr ∗e) override

  Checks if the given expression equals this Var expression.
- int interp () const override

  Evaluates the Var expression. Throws an exception as there's no value for the variable.
- bool has_variable () const override

  Checks if the Var expression has a variable.
- Expr ∗ subst (const std::string &var, Expr ∗replacement) const override

  Substitutes a variable in the Var expression with a replacement expression.
- Expr ∗ clone () const override

  Clones the Var expression.
- void print (std::ostream &os) const override

  Prints the Var expression to the output stream.

Public Member Functions inherited from Expr

- std::string to_string () const

  Returns a string representation of the expression.
- virtual void pretty_print_at (std::ostream &os, precedence_t prec_current) const

  Pretty printsthe expression with varying parentheses.
- virtual void pretty_print (std::ostream &os) const

  calls recursive method pretty_print_at
- virtual std::string to_pretty_string () const

  Returns a pretty string representation of the expression.

Public Attributes

- std::string **varName**

  Represents a varible.

4.5.1 Constructor & Destructor Documentation

4.5.1.1 Var()

```
Var::Var (
            const std::string & varName )
```

Constructs a Var expression with the specified variable name.

Parameters

varName	The name of the variable.

4.5.2 Member Function Documentation

4.5.2.1 clone()

```
Expr * Var::clone ( ) const  [override], [virtual]
```

Clones the Var expression.

Returns

A new copy of the Var expression.

Implements Expr.

4.5.2.2 equals()

```
bool Var::equals (
            Expr * e )  [override], [virtual]
```

Checks if the given expression equals this Var expression.

Parameters

e	The expression to compare with.

Returns

True if the expressions are equal, false otherwise.

Implements Expr.

4.5.2.3 has_variable()

```
bool Var::has_variable ( ) const  [override], [virtual]
```

Checks if the Var expression has a variable.

Returns

True if the Var expression has a variable, false otherwise.

Implements Expr.

4.5.2.4 interp()

```
int Var::interp ( ) const  [override], [virtual]
```

Evaluates the Var expression. Throws an exception as there's no value for the variable.

Returns

This function throws a std::runtime_error.

Implements Expr.

4.5.2.5 print()

```
void Var::print (
            std::ostream & os ) const  [override], [virtual]
```

Prints the Var expression to the output stream.

Parameters

os	The output stream to print to.

Reimplemented from Expr.

4.5.2.6 subst()

```
Expr * Var::subst (
            const std::string & var,
            Expr * replacement ) const  [override], [virtual]
```

Substitutes a variable in the Var expression with a replacement expression.

If the variable name matches the name of this Var expression, it returns a clone of the replacement expression. Otherwise, it returns a new Var expression with the same name as this Var expression.

Parameters

var	The variable to substitute.
replacement	The expression to replace the variable with.

Returns

A new expression after substitution.

Implements Expr.

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

- expr.h
- expr.cpp

Chapter 5

File Documentation

5.1 cmdline.cpp File Reference

```
#include "cmdline.hpp"
#include <stdio.h>
#include "expr.h"
#include "catch.h"
#include <iostream>
#include <cstdlib>
```

Functions

- bool use_arguments (int argc, char ∗argv[ ])

  Parses command-line arguments to determine if tests should be run.

5.1.1 Detailed Description

\Checks user inputs

If user inserts –test all tests run, If they enter –help lists all calls. And returns error if the correct call isnt entered.

Author

Rylie Byers

5.1.2 Function Documentation

5.1.2.1 use_arguments()

```
bool use_arguments (
            int argc,
            char * argv[] )
```

Parses command-line arguments to determine if tests should be run.

Parameters

argc	The number of command-line arguments
argv	The array of command-line arguments.

Returns

bool True if the "--test" flag is present, indicating that tests should be run; false otherwise

5.2 cmdline.hpp File Reference

expression class

```
#include <stdio.h>
```

Functions

- bool use_arguments (int argc, char ∗argv[ ])

  Parses command-line arguments to determine if tests should be run.

5.2.1 Detailed Description

expression class

Checks users input

5.2.2 Function Documentation

5.2.2.1 use_arguments()

```
bool use_arguments (
            int argc,
            char * argv[] )
```

Parses command-line arguments to determine if tests should be run.

Parameters

argc	The number of command-line arguments
argv	The array of command-line arguments.

Returns

bool True if the "--test" flag is present, indicating that tests should be run; false otherwise

5.3 cmdline.hpp

Go to the documentation of this file.

```
00001
00009 #ifndef cmdline_hpp
00010 #define cmdline_hpp
00011
00012 #include <stdio.h>
00013
00014
00015 bool use_arguments(int argc, char* argv[]);
00016
00017 #endif /* cmdline_hpp */
```

5.4 expr.cpp File Reference

Implementation file for the Expr class and its subclasses.

```
#include "expr.h"
#include <iostream>
#include <stdexcept>
#include <sstream>
```

5.4.1 Detailed Description

Implementation file for the Expr class and its subclasses.

This file contains the implementation of the Expr class and its subclasses: Num, Add, Mult, and Var. The Expr class represents an arithmetic expression, and its subclasses represent different types of expressions, such as numbers, addition, multiplication, and variables.

Author

Rylie Byers

Date

Created on January 22, 2024

5.5 expr.h File Reference

brief Defines classes for mathematical expressions.

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

Classes

- class Expr
- class Num
- class Add
- class Mult
- class Var

Enumerations

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

5.5.1 Detailed Description

brief Defines classes for mathematical expressions.

There is the expr class. Add, mult, and var subclasses. All of the expr calls get overridden by Add, mult, and var subclasses

5.6 expr.h

Go to the documentation of this file.

```
00001
00008 #ifndef expr_h
00009 #define expr_h
00010 #pragma once
00011 #include <string>
00012 #include <iostream>
00013
00014 typedef enum {
00015     prec_none, //0
00016     prec_add, //1
00017     prec_mult //1
00018 } precedence_t;
00019
00020 //=========Class for expressions=========
00021 class Expr {
00022
00023 public:
00024     virtual bool equals(Expr *e) = 0;
00025     virtual int interp() const = 0;
00026     virtual bool has_variable() const = 0;
00027     virtual Expr* subst(const std::string& var, Expr* replacement) const = 0;
00028     virtual Expr* clone() const = 0;
00029     virtual ~Expr() {}
00030     std::string to_string() const;
00031     virtual void print(std::ostream& os) const;
00032     virtual void pretty_print_at(std::ostream& os, precedence_t prec_current) const;
00033     virtual void pretty_print(std::ostream& os) const;
00034
00035     virtual std::string to_pretty_string() const;
00036
00037
00038 };
00039
00040 //=========Subclass for number expressions=========
00041 class Num : public Expr {
00042 public:
00043     int val;
00044
00045     Num(int val);
00046
00047     bool equals(Expr* e) override;
00048     int interp() const override;
00049     bool has_variable() const override;
00050     Expr* subst(const std::string& var, Expr* replacement) const override;
00051     Expr* clone() const override;
00052     std::string to_string() const;
00053     void print(std::ostream& os) const override;
00054 };
00055
00056 //========Subclass for addition expressions========
00057 class Add : public Expr {
00058 public:
00059     Expr* lhs;
00060     Expr* rhs;
00061
00062     Add(Expr* lhs, Expr* rhs);
00063
00064     bool equals(Expr* e) override;
00065     int interp() const override;
00066     bool has_variable() const override;
```

```
00067     Expr* subst(const std::string& var, Expr* replacement) const override;
00068     Expr* clone() const override;
00069     void print(std::ostream& os) const override;
00070     void pretty_print(std::ostream& os) const override;
00071 
00072     std::string to_pretty_string() const override;
00073 };
00074 
00075 //========Subclass for multiplication expressions========
00076 class Mult : public Expr {
00077 public:
00078     Expr* lhs;
00079     Expr* rhs;
00080 
00081     Mult(Expr* lhs, Expr* rhs);
00082 
00083     bool equals(Expr* e) override;
00084     int interp() const override;
00085     bool has_variable() const override;
00086     Expr* subst(const std::string& var, Expr* replacement) const override;
00087     Expr* clone() const override;
00088       void print(std::ostream& os) const override;
00089        void pretty_print(std::ostream& os) const override;
00090     std::string to_pretty_string() const override;
00091 };
00092 
00093 //========Subclass for variable expressions========
00094 
00095 class Var : public Expr {
00096 public:
00097     std::string varName;
00098 
00099     Var(const std::string& varName);
00100 
00101     bool equals(Expr* e) override;
00102     int interp() const override;
00103     bool has_variable() const override;
00104     Expr* subst(const std::string& var, Expr* replacement) const override;
00105     Expr* clone() const override;
00106     void print(std::ostream& os) const override;
00107 };
00108 
00109 
00110 #endif /* expr_h */
```

Index