MSDscript

Generated by Doxygen 1.9.6

Chapter 1

MSDScript

Author

Reshma Raghavan

Date

01-11-2023

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

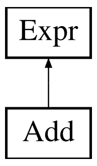
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Add Class Reference

Inheritance diagram for Add:



Public Member Functions

- Add (Expr ∗left, Expr ∗right)

    Constructor.
- bool equals (Expr ∗e)

    Overriding "equals" method in Expr (base) class Basically, we compare two Add objects and determine if they are equal to one another by comparing their member variables (both Expr∗ lhs and Expr∗ rhs)
- int interp ()

    Add returns a number (an int) Basically, interp() interprets the value of the object. In this case, an Add object contains two expressions that should be added to one another; the sum is returned.
- bool has_variable ()

    Add returns true if it HAS a variable The general rule of thumb is this function returns true if the Expression is a variable or has a variable In Add, we check if the lhs or the rhs HAVE a variable.
- Expr ∗ subst (std::string str, Expr ∗e)

    If the first param (str) is in either lhs or rhs, we replace it with e and return.

- virtual bool equals (Expr ∗e)=0

    Expression class: it is an abstract class with virtual methods to be implemented in its derived classes.
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr ∗ subst (std::string str, Expr ∗e)=0

Public Attributes

- Expr ∗ **lhs**
- Expr ∗ **rhs**

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Add()

```
Add::Add (
            Expr ∗ left,
            Expr ∗ right )
```

Constructor.

Parameters

left	a pointer to an Expression
right	a pointer to an Expression

Returns

nothing (it is a constructor so it just creates an object of type Add)

5.1.2 Member Function Documentation

5.1.2.1 equals()

```
bool Add::equals (
            Expr ∗ e )  [virtual]
```

Overriding "equals" method in Expr (base) class Basically, we compare two Add objects and determine if they are equal to one another by comparing their member variables (both Expr∗ lhs and Expr∗ rhs)

Parameters

e	Pointer to an expression

Returns

true or false (it is a boolean)

Implements Expr.

5.1.2.2 has_variable()

```
bool Add::has_variable ( )  [virtual]
```

Add returns true if it HAS a variable The general rule of thumb is this function returns true if the Expression is a variable or has a variable In Add, we check if the lhs or the rhs HAVE a variable.

Parameters

none	

Returns

true or false (it is a boolean)

Implements Expr.

5.1.2.3 interp()

```
int Add::interp ( )  [virtual]
```

Add returns a number (an int) Basically, interp() interprets the value of the object. In this case, an Add object contains two expressions that should be added to one another; the sum is returned.

Parameters

none	

Returns

the sum of two expressions

Implements Expr.

5.1.2.4 subst()

```
Expr * Add::subst (
             std::string str,
             Expr * e )  [virtual]
```

If the first param (str) is in either lhs or rhs, we replace it with e and return.

Parameters

str	a String value
e	a pointer to an Expression

Returns

a new Add Expression

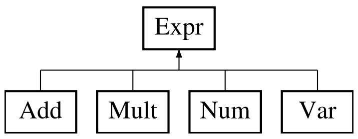
Implements Expr.

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

- /Users/reshmaraghavan/Desktop/msdscript/Expr.h
- /Users/reshmaraghavan/Desktop/msdscript/Expr.cpp

5.2 Expr Class Reference

Inheritance diagram for Expr:



Public Member Functions

- virtual bool equals (Expr ∗e)=0

  Expression class: it is an abstract class with virtual methods to be implemented in its derived classes.
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr ∗ subst (std::string str, Expr ∗e)=0

5.2.1 Member Function Documentation

5.2.1.1 equals()

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

Expression class: it is an abstract class with virtual methods to be implemented in its derived classes.

Implemented in Num, Add, Mult, and Var.

5.2.1.2 has_variable()

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

Implemented in Num, Add, Mult, and Var.

5.2.1.3 interp()

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

Implemented in Num, Add, Mult, and Var.

5.2.1.4 subst()

```
virtual Expr * Expr::subst (
             std::string str,
             Expr * e )  [pure virtual]
```

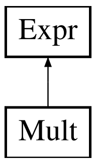
Implemented in Num, Add, Mult, and Var.

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

- /Users/reshmaraghavan/Desktop/msdscript/Expr.h

5.3 Mult Class Reference

Inheritance diagram for Mult:



Public Member Functions

- Mult (Expr ∗left, Expr ∗right)

  Constructor.
- bool equals (Expr ∗e)

  Overriding "equals" method in Expr (base) class Basically, we compare two Mult objects and determine if they are equal to one another by comparing their member variables (both Expr∗ lhs and Expr∗ rhs)
- int interp ()

  Mult returns a number (an int) Basically, interp() interprets the value of the object. In this case, a Mult object contains two expressions that should be multiplied by one another; the product is returned.
- bool has_variable ()

  Mult returns true if it HAS a variable The general rule of thumb is this function returns true if the Expression is a variable or has a variable In Mult, we check if the lhs or the rhs HAVE a variable.
- Expr ∗ subst (std::string str, Expr ∗e)

  If the first param (str) is in either lhs or rhs, we replace it with e and return.

- virtual bool equals (Expr ∗e)=0

  Expression class: it is an abstract class with virtual methods to be implemented in its derived classes.
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr ∗ subst (std::string str, Expr ∗e)=0

Public Attributes

- Expr ∗ **lhs**
- Expr ∗ **rhs**

5.3.1 Constructor & Destructor Documentation

5.3.1.1 Mult()

```
Mult::Mult (
            Expr * left,
            Expr * right )
```

Constructor.

Parameters

left	a pointer to an Expression
right	a pointer to an Expression

Returns

nothing (it is a constructor so it just creates an object of type Add)

5.3.2 Member Function Documentation

5.3.2.1 equals()

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

Overriding "equals" method in Expr (base) class Basically, we compare two Mult objects and determine if they are equal to one another by comparing their member variables (both Expr∗ lhs and Expr∗ rhs)

Parameters

e	Pointer to an expression

Returns

true or false (it is a boolean)

Implements Expr.

5.3.2.2 has_variable()

```
bool Mult::has_variable ( )  [virtual]
```

Mult returns true if it HAS a variable The general rule of thumb is this function returns true if the Expression is a variable or has a variable In Mult, we check if the lhs or the rhs HAVE a variable.

Parameters

none	

Returns

true or false (it is a boolean)

Implements Expr.

5.3.2.3 interp()

```
int Mult::interp ( )  [virtual]
```

Mult returns a number (an int) Basically, interp() interprets the value of the object. In this case, a Mult object contains two expressions that should be multiplied by one another; the product is returned.

Parameters

none	

Returns

the product of two subexpression classes

Implements Expr.

5.3.2.4 subst()

```
Expr * Mult::subst (
             std::string str,
             Expr * e )  [virtual]
```

If the first param (str) is in either lhs or rhs, we replace it with e and return.

Parameters

str	a String value
e	a pointer to an Expression

Returns

a new Mult Expression

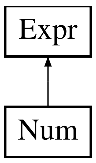
Implements Expr.

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

- /Users/reshmaraghavan/Desktop/msdscript/Expr.h
- /Users/reshmaraghavan/Desktop/msdscript/Expr.cpp

5.4 Num Class Reference

Inheritance diagram for Num:



Public Member Functions

- Num (int value)

  Constructor.
- bool equals (Expr ∗e)

  Overriding "equals" method in Expr (base) class Basically, we compare two Num objects and determine if they are equal to one another by comparing their member variables (int val)
- int interp ()

  Num returns a number (an int) Basically, interp() interprets the value of the object. In this case, a Num object contains an integer; this is returned.
- bool has_variable ()

  Num returns false since it IS NOT a variable and DOES NOT HAVE a variable The general rule of thumb is this function returns true if the Expression is a variable or has a variable.
- Expr ∗ subst (std::string str, Expr ∗e)

  Num will never contain a string, so it just returns the original value.

- virtual bool equals (Expr ∗e)=0

  Expression class: it is an abstract class with virtual methods to be implemented in its derived classes.
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr ∗ subst (std::string str, Expr ∗e)=0

Public Attributes

- int **val**

5.4.1 Constructor & Destructor Documentation

5.4.1.1 Num()

```
Num::Num (
             int value )
```

Constructor.

Parameters

value	an integer value

Returns

nothing (it is a constructor so it just creates an object of type Num)

5.4.2 Member Function Documentation

5.4.2.1 equals()

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

Overriding "equals" method in Expr (base) class Basically, we compare two Num objects and determine if they are equal to one another by comparing their member variables (int val)

Parameters

e	Pointer to an expression

Returns

true or false (it is a boolean)

Implements Expr.

5.4.2.2 has_variable()

```
bool Num::has_variable ( )  [virtual]
```

Num returns false since it IS NOT a variable and DOES NOT HAVE a variable The general rule of thumb is this function returns true if the Expression is a variable or has a variable.

Parameters

none	

Returns

true or false (it is a boolean)

Implements Expr.

5.4.2.3 interp()

```
int Num::interp ( )  [virtual]
```

Num returns a number (an int) Basically, interp() interprets the value of the object. In this case, a Num object contains an integer; this is returned.

Parameters

none	

Returns

int

Implements Expr.

5.4.2.4 subst()

```
Expr * Num::subst (
            std::string str,
            Expr * e )  [virtual]
```

Num will never contain a string, so it just returns the original value.

Parameters

str	a String value
e	a pointer to an Expression

Returns

a new Num Expression (containing the same integer val)

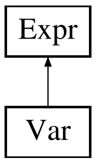
Implements Expr.

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

- /Users/reshmaraghavan/Desktop/msdscript/Expr.h
- /Users/reshmaraghavan/Desktop/msdscript/Expr.cpp

5.5 Var Class Reference

Inheritance diagram for Var:



Public Member Functions

- Var (std::string value)

  Constructor.
- bool equals (Expr ∗e)

  Overriding "equals" method in Expr (base) class Basically, we compare two Var objects and determine if they are equal to one another by comparing their member variables (String val)
- int interp ()

  A Var has no int equivalent.
- bool has_variable ()

  Var returns true as it IS a variable The general rule of thumb is this function returns true if the Expression is a variable or has a variable.
- Expr ∗ subst (std::string str, Expr ∗e)

  If the first param (str) is in the original Var, we replace it with e and return.

- virtual bool equals (Expr ∗e)=0

  Expression class: it is an abstract class with virtual methods to be implemented in its derived classes.
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr ∗ subst (std::string str, Expr ∗e)=0

Public Attributes

- std::string **val**

5.5.1 Constructor & Destructor Documentation

5.5.1.1 Var()

```
Var::Var (
            std::string value )
```

Constructor.

Parameters

value	a String value

Returns

nothing (it is a constructor so it just creates an object of type Var)

5.5.2 Member Function Documentation

5.5.2.1 equals()

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

Overriding "equals" method in Expr (base) class Basically, we compare two Var objects and determine if they are equal to one another by comparing their member variables (String val)

Parameters

e	Pointer to an expression

Returns

true or false (it is a boolean)

Implements Expr.

5.5.2.2 has_variable()

```
bool Var::has_variable ( )  [virtual]
```

Var returns true as it IS a variable The general rule of thumb is this function returns true if the Expression is a variable or has a variable.

Parameters

none	

Returns

true or false (it is a boolean)

Implements Expr.

5.5.2.3 interp()

```
int Var::interp ( )  [virtual]
```

A Var has no int equivalent.

Parameters

none	

Returns

throws a runtime error

Implements Expr.

5.5.2.4 subst()

```
Expr * Var::subst (
            std::string str,
            Expr * e )  [virtual]
```

If the first param (str) is in the original Var, we replace it with e and return.

Parameters

str	a String value
e	a pointer to an Expression

Returns

a new Var Expression

Implements Expr.

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

- /Users/reshmaraghavan/Desktop/msdscript/Expr.h
- /Users/reshmaraghavan/Desktop/msdscript/Expr.cpp

Chapter 6

File Documentation

6.1 /Users/reshmaraghavan/Desktop/msdscript/cmdline.cpp File Reference

contains single method to run executable

```
#include "cmdline.hpp"
#include <string>
#include <iostream>
#include <regex>
#include "catch.h"
```

Functions

- void **use_arguments** (int argc, char **argv)

6.1.1 Detailed Description

contains single method to run executable

Runs executable with 0 or more arguments passed in

Parameters

argc	first argument, number of arguments
argv	second argument, pointer to array of argument values

Returns

nothing (it is a void function)

Author

Reshma Raghavan

Date

01-11-2023

6.2 /Users/reshmaraghavan/Desktop/msdscript/cmdline.hpp File Reference

contains declaration of a single method; more information available in complementary .cpp file

Functions

- void **use_arguments** (int argc, char **argv)

6.2.1 Detailed Description

contains declaration of a single method; more information available in complementary .cpp file

Author

Reshma Raghavan

Date

01-11-2023

6.3 /Users/reshmaraghavan/Desktop/msdscript/cmdline.hpp

Go to the documentation of this file.

```
#pragma once

void use_arguments(int argc, char **argv);
```

6.4 /Users/reshmaraghavan/Desktop/msdscript/Expr.cpp File Reference

contains method implementations of subclasses

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

6.4.1 Detailed Description

contains method implementations of subclasses

Author

Reshma Raghavan

Date

01-17-2023

6.5 /Users/reshmaraghavan/Desktop/msdscript/Expr.h File Reference

contains declaration of the abstract class's methods; more information available in complementary .cpp file

```
#include <string>
```

Classes

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

6.5.1 Detailed Description

contains declaration of the abstract class's methods; more information available in complementary .cpp file

Author

Reshma Raghavan

Date

01-17-2023

6.6 /Users/reshmaraghavan/Desktop/msdscript/Expr.h

Go to the documentation of this file.

```

#ifndef EXPRESSION_CLASSES_EXPR_H
#define EXPRESSION_CLASSES_EXPR_H


#include <string>

class Expr {

public:
    virtual bool equals(Expr* e) = 0;
    virtual int interp() = 0;
    virtual bool has_variable() = 0;
    virtual Expr* subst (std::string str, Expr* e) = 0;
};

class Num : public Expr {
public:
    int val;
    Num(int value);
    bool equals(Expr *e);
    int interp();
    bool has_variable();
    Expr* subst (std::string str, Expr* e);
};

class Add : public Expr {
public:
    Expr* lhs;
    Expr* rhs;
    Add(Expr* left, Expr* right);
    bool equals(Expr *e);
    int interp();
    bool has_variable();
    Expr* subst (std::string str, Expr* e);
};

class Mult: public Expr {
public:
    Expr* lhs;
    Expr* rhs;
    Mult(Expr* left, Expr* right);
    bool equals(Expr *e);
    int interp();
    bool has_variable();
    Expr* subst (std::string str, Expr* e);
};

class Var: public Expr {
public:
    std::string val;
    Var(std::string value);
    bool equals(Expr *e);
    int interp();
    bool has_variable();
    Expr* subst (std::string str, Expr* e);
};



#endif //EXPRESSION_CLASSES_EXPR_H
```

6.7 /Users/reshmaraghavan/Desktop/msdscript/TestExpr.cpp File Reference

contains test cases of Expression subclass methods

```
#include "catch.h"
#include "Expr.h"
```

Functions

- TEST_CASE ("equals")
- TEST_CASE ("interp")
- TEST_CASE ("has_variable")
- TEST_CASE ("subst")

6.7.1 Detailed Description

contains test cases of Expression subclass methods

Author

Reshma Raghavan

Date

01-23-2023

6.7.2 Function Documentation

6.7.2.1 TEST_CASE() [1/4]

```
TEST_CASE (
              "equals"  )
```

Testing the "equals" method. Asserting true or false as the case may be.

6.7.2.2 TEST_CASE() [2/4]

```
TEST_CASE (
              "has_variable"  )
```

Testing the "has_variable" method. Asserting true or false as the case may be.

6.7.2.3 TEST_CASE() [3/4]

```
TEST_CASE (
              "interp"  )
```

Testing the "interp" method. Asserting true or false as the case may be.

6.7.2.4 TEST_CASE() [4/4]

```
TEST_CASE (
              "subst"  )
```

Testing the "subst" method. Asserting true or false as the case may be.

Index