msdscript

Generated by Doxygen 1.9.6

Chapter 1

MSDScript

Author

Jinny

Date

01-17-2023

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

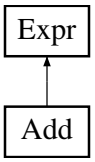
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Add Class Reference

Inheritance diagram for Add:



Public Member Functions

- **Add** (Expr *lhs, Expr *rhs)
- bool equals (Expr *e)

  Compare if two Expressions are the same type with same arguments.
- int interp ()

  Calculate the sum of the subexpression values.
- bool has_variable ()

  Check if the Add expression is a variable or contains a variable.
- Expr * subst (std::string str_new, Expr *e)

  Everywhere that the expression (whose subst method is called) contains a variable matching the str_new, the result Expr should have the given replacement, instead.*

- virtual bool equals (Expr *e)=0
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr * subst (std::string str_new, Expr *e)=0

Public Attributes

- Expr * **lhs**

  left-hand-side expression of the Number expression
- Expr * **rhs**

  right-hand-side expression of the Number expression

5.1.1 Member Function Documentation

5.1.1.1 equals()

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

Compare if two Expressions are the same type with same arguments.

Parameters

e	first argument,the Expression to be compared

Returns

bool if the Expression to be compared is a Add Expression with same arguments

Implements Expr.

5.1.1.2 has_variable()

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

Check if the Add expression is a variable or contains a variable.

Returns

bool if the Add expression is a variable or contains a variable.

Implements Expr.

5.1.1.3 interp()

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

Calculate the sum of the subexpression values.

Returns

int for this Add Expression

Implements Expr.

5.1.1.4 subst()

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

Everywhere that the expression (whose subst method is called) contains a variable matching the str_new, the result Expr∗ should have the given replacement, instead.

Parameters

str_new	first argument, the string to be replaced
e	second argument, the Expression replace str_new

Returns

Expr* that after replacement

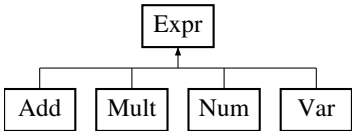
Implements Expr.

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

- /Users/jinny/MSD/my6015Repo/expr.hpp
- /Users/jinny/MSD/my6015Repo/expr.cpp

5.2 Expr Class Reference

Inheritance diagram for Expr:



Public Member Functions

- virtual bool equals (Expr *e)=0
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr * subst (std::string str_new, Expr *e)=0

5.2.1 Member Function Documentation

5.2.1.1 equals()

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

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_new,
            Expr * e )  [pure virtual]
```

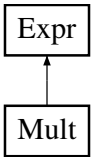
Implemented in Num, Add, Mult, and Var.

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

- /Users/jinny/MSD/my6015Repo/expr.hpp

5.3 Mult Class Reference

Inheritance diagram for Mult:



Public Member Functions

- **Mult** (Expr *lhs, Expr *rhs)
- bool equals (Expr *e)

  Compare if two Expressions are the same type with same arguments.
- int interp ()

  Calculate the product of the subexpression values.
- bool has_variable ()

  Check if the Mult expression is a variable or contains a variable.
- Expr * subst (std::string str_new, Expr *e)

  Everywhere that the expression (whose subst method is called) contains a variable matching the str_new, the result Expr should have the given replacement, instead.*

- virtual bool equals (Expr *e)=0
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr * subst (std::string str_new, Expr *e)=0

Public Attributes

- Expr * **lhs**

  left-hand-side expression of the Mult expression
- Expr * **rhs**

  right-hand-side expression of the Mult expression

5.3.1 Member Function Documentation

5.3.1.1 equals()

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

Compare if two Expressions are the same type with same arguments.

Parameters

e	first argument,the Expression to be compared

Returns

bool if the Expression to be compared is a Mult Expression with same arguments

Implements Expr.

5.3.1.2 has_variable()

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

Check if the Mult expression is a variable or contains a variable.

Returns

bool if the Mult expression is a variable or contains a variable.

Implements Expr.

5.3.1.3 interp()

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

Calculate the product of the subexpression values.

Returns

int for this Mult Expression

Implements Expr.

5.3.1.4 subst()

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

Everywhere that the expression (whose subst method is called) contains a variable matching the str_new, the result Expr* should have the given replacement, instead.

Parameters

str_new	first argument, the string to be replaced
e	second argument, the Expression replace str_new

Returns

Expr* that after replacement

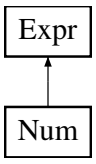
Implements Expr.

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

- /Users/jinny/MSD/my6015Repo/expr.hpp
- /Users/jinny/MSD/my6015Repo/expr.cpp

5.4 Num Class Reference

Inheritance diagram for Num:



Public Member Functions

- **Num** (int val)
- bool equals (Expr *e)

  Compare if two Expressions are the same type with same arguments.
- int interp ()

  return an int for the value of the number.
- bool has_variable ()

  Check if the expression is a variable or contains a variable. Num expression always return false.
- Expr * subst (std::string str_new, Expr *e)

  Everywhere that the expression (whose subst method is called) contains a variable matching the str_new, the result Expr should have the given replacement, instead.*

- virtual bool equals (Expr *e)=0
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr * subst (std::string str_new, Expr *e)=0

Public Attributes

- int **val**

  integer value of the Number expression

5.4.1 Member Function Documentation

5.4.1.1 equals()

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

Compare if two Expressions are the same type with same arguments.

Parameters

e	first argument,the Expression to be compared

Returns

bool if the Expression to be compared is a Number Expression

Implements Expr.

5.4.1.2 has_variable()

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

Check if the expression is a variable or contains a variable. Num expression always return false.

Returns

bool if the num expression is a variable. It should always return false.

Implements Expr.

5.4.1.3 interp()

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

return an int for the value of the number.

Returns

int for this Number Expression

Implements Expr.

5.4.1.4 subst()

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

Everywhere that the expression (whose subst method is called) contains a variable matching the str_new, the result Expr* should have the given replacement, instead.

Parameters

str_new	first argument, the string to be replaced
e	second argument, the Expression replace str_new

Returns

Expr* that after replacement

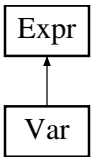
Implements Expr.

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

- /Users/jinny/MSD/my6015Repo/expr.hpp
- /Users/jinny/MSD/my6015Repo/expr.cpp

5.5 Var Class Reference

Inheritance diagram for Var:



Public Member Functions

- **Var** (std::string str)
- bool equals (Expr ∗e)

  Compare if two Expressions are the same type with same arguments.
- int interp ()

  A variable has no value, so interp for a variable should throw a std::runtime_error exception.
- bool has_variable ()

  Check if the Var expression is a variable.
- Expr ∗ subst (std::string str_new, Expr ∗e)

  Everywhere that the expression (whose subst method is called) contains a variable matching the str_new, the result Expr∗ should have the given replacement, instead.

- virtual bool equals (Expr ∗e)=0
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr ∗ subst (std::string str_new, Expr ∗e)=0

Public Attributes

- std::string **str**

  string value of the Var expression

5.5.1 Member Function Documentation

5.5.1.1 equals()

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

Compare if two Expressions are the same type with same arguments.

Parameters

e	first argument,the Expression to be compared

Returns

bool if the Expression to be compared is a Var Expression with same arguments

Implements Expr.

5.5.1.2 has_variable()

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

Check if the Var expression is a variable.

Returns

bool if the Var expression is a variable. It should always return true.

Implements Expr.

5.5.1.3 interp()

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

A variable has no value, so interp for a variable should throw a std::runtime_error exception.

Implements Expr.

5.5.1.4 subst()

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

Everywhere that the expression (whose subst method is called) contains a variable matching the str_new, the result Expr* should have the given replacement, instead.

Parameters

str_new	first argument, the string to be replaced
e	second argument, the Expression replace str_new

Returns

Expr* that after replacement

Implements Expr.

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

- /Users/jinny/MSD/my6015Repo/expr.hpp
- /Users/jinny/MSD/my6015Repo/expr.cpp

Chapter 6

File Documentation

6.1 /Users/jinny/MSD/my6015Repo/cmdline.cpp File Reference

contains cmdline class definition

```
#include "cmdline.hpp"
```

Functions

- void **use_arguments** (int argc, const char ∗argv[ ])

6.1.1 Detailed Description

contains cmdline class definition

execution for each command line argument

Author

Jinny

6.2 /Users/jinny/MSD/my6015Repo/cmdline.hpp

```
00001
00008 #ifndef cmdline_hpp
00009 #define cmdline_hpp
00010
00011 #include <stdio.h>
00012 #include <iostream>
00013
00014 #include "catch.h"
00015
00016 void use_arguments(int argc, const char * argv[]);
00017
00018 #endif /* cmdline_hpp */
```

6.3 /Users/jinny/MSD/my6015Repo/expr.cpp File Reference

contains expression class definition

```
#include "expr.hpp"
#include "catch.h"
```

Functions

- **TEST_CASE** ("Test_Equal_Within_Same_Class")
- **TEST_CASE** ("Test_Equal_Not_Same_Class")
- **TEST_CASE** ("Test_interp()")
- **TEST_CASE** ("Test_hasVariable()")
- **TEST_CASE** ("Test_subst()")

6.3.1 Detailed Description

contains expression class definition

No more verbose description here lol

Author

Jinny

6.4 /Users/jinny/MSD/my6015Repo/expr.hpp

```
00001
00008 #ifndef expr_hpp
00009 #define expr_hpp
00010
00011 #include <stdio.h>
00012 #include <string>
00013 #include <stdexcept>
00014
00015 class Expr {
00016 public:
00017     virtual bool equals(Expr *e) = 0;
00018     virtual int interp() = 0;
00019     virtual bool has_variable() = 0;
00020     virtual Expr* subst(std::string str_new, Expr *e) = 0;
00021 //    virtual void print(std::ostream&) = 0;
00022 //    std::string to_string();
00023 //    virtual void pretty_print(std::ostream&) = 0;
00024 };
00025
00026 class Num: public Expr {
00027 public:
00028     int val;
00029     Num(int val);
00030     bool equals(Expr *e);
00031     int interp();
00032     bool has_variable();
00033     Expr* subst(std::string str_new, Expr *e);
00034 //    void print(std::ostream&);
00035 //    std::string to_string();
00036 };
00037
00038
00039 class Add: public Expr {
00040 public:
00041     Expr *lhs;
00042     Expr *rhs;
00043     Add(Expr *lhs, Expr *rhs);
```

```
00044     bool equals(Expr *e);
00045     int interp();
00046     bool has_variable();
00047     Expr* subst(std::string str_new, Expr *e);
00048 //    void print(std::ostream&);
00049 //    std::string to_string();
00050 };
00051
00052
00053 class Mult: public Expr {
00054 public:
00055     Expr *lhs;
00056     Expr *rhs;
00057     Mult(Expr *lhs, Expr *rhs);
00058     bool equals(Expr *e);
00059     int interp();
00060     bool has_variable();
00061     Expr* subst(std::string str_new, Expr *e);
00062 //    void print(std::ostream&);
00063 //    std::string to_string();
00064 };
00065
00066
00067 class Var: public Expr {
00068 public:
00069     std::string str;
00070     Var(std::string str);
00071     bool equals(Expr *e);
00072     int interp();
00073     bool has_variable();
00074     Expr* subst(std::string str_new, Expr *e);
00075 //    void print(std::ostream&);
00076 //    std::string to_string();
00077 };
00078
00079
00080
00081
00082 #endif /* expr_hpp */
```

Index