# MSDScript

Generated by Doxygen 1.9.6

# Chapter 1

# MSDScript

**Author**

Jowie Tan

**Date**

02-07-2023

# 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

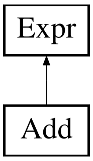
Here is a list of all documented files with brief descriptions:

# Chapter 5

# Class Documentation

## 5.1 Add Class Reference

Inheritance diagram for Add:



### Public Member Functions

- Add (Expr ∗lhs, Expr ∗rhs)

    *Sets value to add expression.*
- bool equals (Expr ∗other)

    *Compare Value.*
- int interp ()

    *Interprets the Add expression.*
- bool has_variable ()

    *check if expression has variable*
- Expr ∗ subst (std::string target, Expr ∗other)

    *if variable matches target string, replaces variable with Expr other. This goes into each expression in each side*

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

### 5.1.1 Constructor & Destructor Documentation

#### 5.1.1.1 Add()

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

Sets value to add expression.

**Parameters**

|  |  |
|---|---|
| *Expr* | left value |
| *Expr* | right value |

## 5.1.2 Member Function Documentation

### 5.1.2.1 equals()

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

Compare Value.

**Parameters**

|  |  |
|---|---|
| *Expr* | other |

**Returns**

Boolean result

Implements Expr.

### 5.1.2.2 has_variable()

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

check if expression has variable

**Returns**

Boolean results

Implements Expr.

#### 5.1.2.3 interp()

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

Interprets the Add expression.

**Returns**

Int add the value of the left expression and right expression

Implements Expr.

#### 5.1.2.4 subst()

```
Expr * Add::subst (
             std::string target,
             Expr * other )  [virtual]
```

if variable matches target string, replaces variable with Expr other. This goes into each expression in each side

**Parameters**

| | |
|---|---|
| *string* | target - target string to replace |
| *Expr∗* | other - replacement expression |

**Returns**

new Add expression

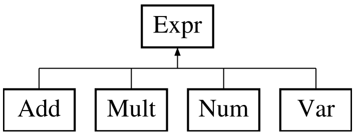
Implements Expr.

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

- /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/expr.h
- /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/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 target, Expr ∗other)=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 target,
             Expr * other )  [pure virtual]
```

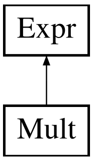
Implemented in Num, Add, Mult, and Var.

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

- /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/expr.h

## 5.3 Mult Class Reference

Inheritance diagram for Mult:



### Public Member Functions

- Mult (Expr ∗lhs, Expr ∗rhs)

    *Sets value to mult expression.*
- bool equals (Expr ∗other)

    *Compare Value.*
- int interp ()

    *Interprets the Mult expression.*
- bool has_variable ()

    *check if expression has variable*
- Expr ∗ subst (std::string target, Expr ∗other)

    *if variable matches target string, replaces variable with Expr other. This goes into each expression in each side*

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

### 5.3.1 Constructor & Destructor Documentation

#### 5.3.1.1 Mult()

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

Sets value to mult expression.

**Parameters**

| | |
|---|---|
| *Expr*∗ | left value |
| *Expr*∗ | right value |

## 5.3.2 Member Function Documentation

### 5.3.2.1 equals()

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

Compare Value.

**Parameters**

| | |
|---|---|
| *Expr∗* | other |

**Returns**

Boolean result

Implements Expr.

### 5.3.2.2 has_variable()

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

check if expression has variable

**Returns**

Boolean results

Implements Expr.

### 5.3.2.3 interp()

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

Interprets the Mult expression.

**Returns**

Int times the value of the left expression and right expression

Implements Expr.

### 5.3.2.4 subst()

```
Expr * Mult::subst (
            std::string target,
            Expr * other )  [virtual]
```

if variable matches target string, replaces variable with Expr other. This goes into each expression in each side

**Parameters**

| | |
|---|---|
| *string* | target - target string to replace |
| *Expr∗* | other - replacement expression |

**Returns**

new Mult expression

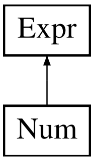
Implements Expr.

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

- /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/expr.h
- /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/expr.cpp

## 5.4 Num Class Reference

Inheritance diagram for Num:



### Public Member Functions

- Num (int val)

  *Sets value.*
- bool equals (Expr ∗other)

  *Compare Value.*
- int interp ()

  *Interprets the Num expression.*
- bool has_variable ()

  *check if expression has variable*
- Expr ∗ subst (std::string target, Expr ∗other)

  *if variable matches target string, replaces variable with Expr other. However, since this is not a variable, there is no change.*

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

### 5.4.1 Constructor & Destructor Documentation

#### 5.4.1.1 Num()

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

Sets value.

**Parameters**

| | |
|---|---|
| *value* | |

## 5.4.2 Member Function Documentation

### 5.4.2.1 equals()

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

Compare Value.

**Parameters**

| | |
|---|---|
| *Expr* | other |

**Returns**

Boolean result

Implements Expr.

### 5.4.2.2 has_variable()

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

check if expression has variable

**Returns**

Boolean false

Implements Expr.

### 5.4.2.3 interp()

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

Interprets the Num expression.

**Returns**

int Value

Implements Expr.

### 5.4.2.4 subst()

```
Expr * Num::subst (
            std::string target,
            Expr * other )  [virtual]
```

if variable matches target string, replaces variable with Expr other. However, since this is not a variable, there is no change.

**Parameters**

| | |
|---|---|
| *string* | target - target string to replace |
| *Expr∗* | other - replacement expression |

**Returns**

this (no change)

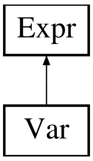
Implements Expr.

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

- /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/expr.h
- /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/expr.cpp

## 5.5 Var Class Reference

Inheritance diagram for Var:



### Public Member Functions

- Var (std::string value)

  *Sets String.*
- bool equals (Expr ∗other)

  *Compare Value.*
- int interp ()

  *Interprets the variable expression.*
- bool has_variable ()

  *check if expression has variable*
- Expr ∗ subst (std::string target, Expr ∗other)

  *if variable matches target string, replaces variable with Expr other*

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

## 5.5.1 Constructor & Destructor Documentation

### 5.5.1.1 Var()

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

Sets String.

**Parameters**

| | |
|---|---|
| *string* | value |

## 5.5.2 Member Function Documentation

### 5.5.2.1 equals()

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

Compare Value.

**Parameters**

| | |
|---|---|
| *Expr∗* | other |

**Returns**

Boolean result

Implements Expr.

### 5.5.2.2 has_variable()

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

check if expression has variable

**Returns**

Boolean true

Implements Expr.

#### 5.5.2.3 interp()

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

Interprets the variable expression.

**Returns**

Runtime error as there is no value to interpret

Implements Expr.

#### 5.5.2.4 subst()

```
Expr * Var::subst (
             std::string target,
             Expr * other )  [virtual]
```

if variable matches target string, replaces variable with Expr other

**Parameters**

| | |
|---|---|
| *string* | target - target string to replace |
| *Expr∗* | other - replacement expression |

**Returns**

if variable match, replace variable with Expr other; if not match, return variable

Implements Expr.

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

- /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/expr.h
- /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/expr.cpp

# Chapter 6

# File Documentation

## 6.1 /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/cmdline.h File Reference

Contain functions to run program from terminal.

```
#include "catch.h"
#include <iostream>
#include <cstdlib>
```

### Functions

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

### 6.1.1 Detailed Description

Contain functions to run program from terminal.

## 6.2 /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/cmdline.h

Go to the documentation of this file.

```

#ifndef CMDLINE_H
#define CMDLINE_H

#define CATCH_CONFIG_RUNNER
#include "catch.h"
#include <iostream>
#include <cstdlib>

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

#endif
```

## 6.3 /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/expr.cpp File Reference

Contains all the expression methods that are inherited by Num, Add, Mult and Var.

```
#include "expr.h"
```

### 6.3.1 Detailed Description

Contains all the expression methods that are inherited by Num, Add, Mult and Var.

## 6.4 /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/expr.h File Reference

Contains the methods that would be defined in expr.cpp.

```
#include <iostream>
#include "catch.h"
```

### Classes

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

### 6.4.1 Detailed Description

Contains the methods that would be defined in expr.cpp.

## 6.5 /Users/jowietan/Desktop/MSD/CS6015/Homework/ExpressionClass/ExpressionClass/expr.h

Go to the documentation of this file.

```

#ifndef EXPR_H
#define EXPR_H

#include <iostream>
#include "catch.h"

class Expr {
    public:
        virtual bool equals(Expr *e) = 0;
        virtual int interp() = 0;
        virtual bool has_variable() = 0;
        virtual Expr* subst(std::string target, Expr* other) = 0;
```

```cpp
};

class Num : public Expr {
    private:
        int val;
    public:
        Num (int val);
        bool equals(Expr *other);
        int interp();
        bool has_variable();
        Expr* subst(std::string target, Expr* other);
};

class Add : public Expr {
    private:
        Expr *lhs;
        Expr *rhs;
    public:
        Add(Expr *lhs, Expr *rhs);
        bool equals(Expr *other);
        int interp();
        bool has_variable();
        Expr* subst(std::string target, Expr* other);
};

class Mult : public Expr {
    private:
        Expr *lhs;
        Expr *rhs;
    public:
        Mult(Expr *lhs, Expr *rhs);
        bool equals(Expr *other);
        int interp();
        bool has_variable();
        Expr* subst(std::string target, Expr* other);
};

class Var : public Expr {
    private:
        std::string variable;
    public:
        Var(std::string value);
        bool equals(Expr *other);
        int interp();
        bool has_variable();
        Expr* subst(std::string target, Expr* other);
};

#endif
```

# Index