

MSDScript

Generated by Doxygen 1.9.6



<b>1 MSDScript</b>	<b>1</b>
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Class Documentation</b>	<b>9</b>
5.1 AddExpr Class Reference	9
5.1.1 Detailed Description	10
5.1.2 Constructor & Destructor Documentation	10
5.1.2.1 AddExpr() [1/5]	10
5.1.2.2 AddExpr() [2/5]	10
5.1.2.3 AddExpr() [3/5]	11
5.1.2.4 AddExpr() [4/5]	11
5.1.2.5 AddExpr() [5/5]	11
5.1.3 Member Function Documentation	11
5.1.3.1 equals()	12
5.1.3.2 has_variable()	12
5.1.3.3 interp()	12
5.1.3.4 pretty_print()	12
5.1.3.5 print()	13
5.1.3.6 subst()	13
5.2 Expr Class Reference	14
5.2.1 Detailed Description	14
5.2.2 Member Function Documentation	14
5.2.2.1 equals()	14
5.2.2.2 has_variable()	15
5.2.2.3 interp()	15
5.2.2.4 pretty_print()	15
5.2.2.5 print()	15
5.2.2.6 subst()	15
5.2.2.7 to_pretty_string()	16
5.2.2.8 to_string()	16
5.3 MultExpr Class Reference	16
5.3.1 Detailed Description	17
5.3.2 Constructor & Destructor Documentation	17
5.3.2.1 MultExpr() [1/5]	17
5.3.2.2 MultExpr() [2/5]	17
5.3.2.3 MultExpr() [3/5]	18

5.3.2.4 MultExpr() [4/5]	18
5.3.2.5 MultExpr() [5/5]	18
5.3.3 Member Function Documentation	19
5.3.3.1 equals()	19
5.3.3.2 has_variable()	19
5.3.3.3 interp()	20
5.3.3.4 pretty_print()	20
5.3.3.5 print()	20
5.3.3.6 subst()	20
5.4 NumExpr Class Reference	22
5.4.1 Detailed Description	23
5.4.2 Constructor & Destructor Documentation	23
5.4.2.1 NumExpr()	23
5.4.3 Member Function Documentation	23
5.4.3.1 equals()	23
5.4.3.2 has_variable()	24
5.4.3.3 interp()	24
5.4.3.4 pretty_print()	24
5.4.3.5 print()	25
5.4.3.6 subst()	25
5.5 VarExpr Class Reference	25
5.5.1 Detailed Description	26
5.5.2 Constructor & Destructor Documentation	26
5.5.2.1 VarExpr()	26
5.5.3 Member Function Documentation	27
5.5.3.1 equals()	27
5.5.3.2 has_variable()	27
5.5.3.3 interp()	27
5.5.3.4 pretty_print()	27
5.5.3.5 print()	28
5.5.3.6 subst()	28
<b>6 File Documentation</b>	<b>29</b>
6.1 /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/cmdline.h	29
6.2 /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/expr.hpp	29
<b>Index</b>	<b>31</b>

# Chapter 1

## MSDScript

Author

Yue Sun

Date

02-07-2023



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

Expr . . . . .	14
AddExpr . . . . .	9
MultExpr . . . . .	16
NumExpr . . . . .	22
VarExpr . . . . .	25





## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">AddExpr</a>	<a href="#">AddExpr</a> class - for add expressions . . . . .	<a href="#">9</a>
<a href="#">Expr</a>	<a href="#">Expr</a> class - the base class for all expressions . . . . .	<a href="#">14</a>
<a href="#">MultExpr</a>	<a href="#">MultExpr</a> class - for multiplication expressions . . . . .	<a href="#">16</a>
<a href="#">NumExpr</a>	<a href="#">NumExpr</a> class - for number expressions . . . . .	<a href="#">22</a>
<a href="#">VarExpr</a>	<a href="#">VarExpr</a> class - for variable expressions . . . . .	<a href="#">25</a>



## Chapter 4

# File Index

### 4.1 File List

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

<a href="#">/Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/cmdline.h</a>	. . . . .	??
<a href="#">/Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/expr.hpp</a>	. . . . .	??



## Chapter 5

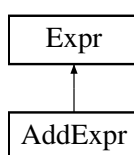
# Class Documentation

### 5.1 AddExpr Class Reference

[AddExpr](#) class - for add expressions.

```
#include <expr.hpp>
```

Inheritance diagram for AddExpr:



#### Public Member Functions

- [AddExpr](#) ([Expr](#) \*left, [Expr](#) \*right)
- [AddExpr](#) (int left, int right)
- [AddExpr](#) (std::string left, int right)
- [AddExpr](#) (int left, std::string right)
- [AddExpr](#) (std::string left, std::string right)
- bool [equals](#) ([Expr](#) \*expr)
- int [interp](#) ()
- bool [has\\_variable](#) ()
- [Expr](#) \* [subst](#) (std::string s, [Expr](#) \*expr)
- void [print](#) (std::ostream &out)
- void [pretty\\_print](#) (std::ostream &out)

#### Public Member Functions inherited from [Expr](#)

- virtual bool [equals](#) ([Expr](#) \*expr)=0
- virtual int [interp](#) ()=0
- virtual bool [has\\_variable](#) ()=0
- virtual [Expr](#) \* [subst](#) (std::string s, [Expr](#) \*expr)=0
- virtual void [print](#) (std::ostream &out)=0
- virtual void [pretty\\_print](#) (std::ostream &out)=0
- std::string [to\\_string](#) ()
- precedence\_t [get\\_precedence](#) ()
- std::string [to\\_pretty\\_string](#) ()

## Additional Inherited Members

### Public Attributes inherited from [Expr](#)

- precedence\_t **prec**

### 5.1.1 Detailed Description

[AddExpr](#) class - for add expressions.

### 5.1.2 Constructor & Destructor Documentation

#### 5.1.2.1 AddExpr() [1/5]

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

Constructs a [AddExpr](#) object with the specified left and right expressions.

##### Parameters

<i>left</i>	the specified left expression
<i>right</i>	the specified right expression

#### 5.1.2.2 AddExpr() [2/5]

```
AddExpr::AddExpr (
    int left,
    int right )
```

Constructs a [AddExpr](#) object with the specified int values.

##### Parameters

<i>left</i>	the specified int value in the left expression
<i>right</i>	the specified int value in the right expression

### 5.1.2.3 AddExpr() [3/5]

```
AddExpr::AddExpr (
    std::string left,
    int right )
```

Constructs a [AddExpr](#) object with a specified string as the left expression and a specified int value as the right expression.

#### Parameters

<i>left</i>	the specified string in the left expression
<i>right</i>	the specified int value in the right expression

### 5.1.2.4 AddExpr() [4/5]

```
AddExpr::AddExpr (
    int left,
    std::string right )
```

Constructs a [AddExpr](#) object with a specified int as the left expression and a specified string value as the right expression.

#### Parameters

<i>left</i>	the specified int value in the left expression
<i>right</i>	the specified string in the right expression

### 5.1.2.5 AddExpr() [5/5]

```
AddExpr::AddExpr (
    std::string left,
    std::string right )
```

Constructs a [AddExpr](#) object with the specified strings.

#### Parameters

<i>left</i>	the specified string in the left <a href="#">VarExpr</a> expression
<i>right</i>	the specified string in the right <a href="#">VarExpr</a> expression

## 5.1.3 Member Function Documentation

### 5.1.3.1 equals()

```
bool AddExpr::equals (
    Expr * expr ) [virtual]
```

Compared the specified expression with this object for equality.

#### Parameters

<i>expr</i>	the expression to be compared for equality with this object
-------------	---

#### Returns

true if the specified expression is a [AddExpr](#) object and it has the same value with this object. Otherwise returns false.

Implements [Expr](#).

### 5.1.3.2 has\_variable()

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

Check if this [AddExpr](#) expression object contains a [VarExpr](#) object.

#### Returns

true if either its left or right expression contains a [VarExpr](#) object

Implements [Expr](#).

### 5.1.3.3 interp()

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

Add the left and right expressions and returns the int value of the result.

#### Returns

the int value of the addition of the left and right expressions

Implements [Expr](#).

### 5.1.3.4 pretty\_print()

```
void AddExpr::pretty_print (
    std::ostream & out ) [virtual]
```

Print this object in a prettier way. To specify, we'll omit redundant parentheses and add a space around operators.



**Parameters**

<i>out</i>	the output stream used to print this object
------------	---

Implements [Expr](#).

**5.1.3.5 print()**

```
void AddExpr::print (
    std::ostream & out ) [virtual]
```

Print this object with parentheses and a add sign.

**Parameters**

<i>out</i>	the output stream used to print this object
------------	---

Implements [Expr](#).

**5.1.3.6 subst()**

```
Expr * AddExpr::subst (
    std::string s,
    Expr * expr ) [virtual]
```

Substitute the specified string in this object with a specified expression

**Parameters**

<i>s</i>	the specified string to be substituted
<i>expr</i>	the specified expression used to substitute the string

**Returns**

a new [AddExpr](#) object with the specified strings in its left and right expressions have been substituted

Implements [Expr](#).

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

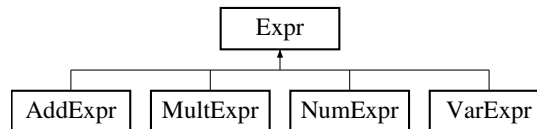
- /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/expr.hpp
- /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/expr.cpp

## 5.2 Expr Class Reference

[Expr](#) class - the base class for all expressions.

```
#include <expr.hpp>
```

Inheritance diagram for Expr:



### Public Member Functions

- virtual bool [equals](#) ([Expr](#) \*expr)=0
- virtual int [interp](#) ()=0
- virtual bool [has\\_variable](#) ()=0
- virtual [Expr](#) \* [subst](#) (std::string s, [Expr](#) \*expr)=0
- virtual void [print](#) (std::ostream &out)=0
- virtual void [pretty\\_print](#) (std::ostream &out)=0
- std::string [to\\_string](#) ()
- precedence\_t [get\\_precedence](#) ()
- std::string [to\\_pretty\\_string](#) ()

### Public Attributes

- precedence\_t [prec](#)

### 5.2.1 Detailed Description

[Expr](#) class - the base class for all expressions.

### 5.2.2 Member Function Documentation

#### 5.2.2.1 equals()

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

Implemented in [NumExpr](#), [AddExpr](#), [MultExpr](#), and [VarExpr](#).

### 5.2.2.2 has\_variable()

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

Implemented in [NumExpr](#), [AddExpr](#), [MultExpr](#), and [VarExpr](#).

### 5.2.2.3 interp()

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

Implemented in [NumExpr](#), [AddExpr](#), [MultExpr](#), and [VarExpr](#).

### 5.2.2.4 pretty\_print()

```
virtual void Expr::pretty_print (
    std::ostream & out ) [pure virtual]
```

Implemented in [NumExpr](#), [AddExpr](#), [MultExpr](#), and [VarExpr](#).

### 5.2.2.5 print()

```
virtual void Expr::print (
    std::ostream & out ) [pure virtual]
```

Implemented in [NumExpr](#), [AddExpr](#), [MultExpr](#), and [VarExpr](#).

### 5.2.2.6 subst()

```
virtual Expr * Expr::subst (
    std::string s,
    Expr * expr ) [pure virtual]
```

Implemented in [NumExpr](#), [AddExpr](#), [MultExpr](#), and [VarExpr](#).

### 5.2.2.7 to\_pretty\_string()

```
std::string Expr::to_pretty_string ( )
```

Store the result of pretty\_print method into a string.

#### Returns

the string representation of the pretty\_print method

### 5.2.2.8 to\_string()

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

Store the result of print method into a string.

#### Returns

the string representation of the print method

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

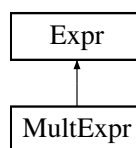
- /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/expr.hpp
- /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/expr.cpp

## 5.3 MultExpr Class Reference

[MultExpr](#) class - for multiplication expressions.

```
#include <expr.hpp>
```

Inheritance diagram for MultExpr:



### Public Member Functions

- [MultExpr](#) ([Expr](#) \*left, [Expr](#) \*right)
- [MultExpr](#) (int left, int right)
- [MultExpr](#) (std::string left, int right)
- [MultExpr](#) (int left, std::string right)
- [MultExpr](#) (std::string left, std::string right)
- bool [equals](#) ([Expr](#) \*expr)
- int [interp](#) ()
- bool [has\\_variable](#) ()
- [Expr](#) \* [subst](#) (std::string s, [Expr](#) \*expr)
- void [print](#) (std::ostream &out)
- void [pretty\\_print](#) (std::ostream &out)

**Public Member Functions inherited from Expr**

- virtual bool `equals` (Expr \*expr)=0
- virtual int `interp` ()=0
- virtual bool `has_variable` ()=0
- virtual Expr \* `subst` (std::string s, Expr \*expr)=0
- virtual void `print` (std::ostream &out)=0
- virtual void `pretty_print` (std::ostream &out)=0
- std::string `to_string` ()
- precedence\_t `get_precedence` ()
- std::string `to_pretty_string` ()

**Public Attributes**

- Expr \* `lhs`
- Expr \* `rhs`

**Public Attributes inherited from Expr**

- precedence\_t `prec`

**5.3.1 Detailed Description**

`MultExpr` class - for multiplication expressions.

**5.3.2 Constructor & Destructor Documentation****5.3.2.1 MultExpr() [1/5]**

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

Constructs a `MultExpr` object with the specified left and right expressions.

**Parameters**

<i>left</i>	the specified left expression
<i>right</i>	the specified right expression

**5.3.2.2 MultExpr() [2/5]**

```
MultExpr::MultExpr (
```

```
int left,
int right )
```

Constructs a [MultExpr](#) object with the specified int values.

#### Parameters

<i>left</i>	the specified int value in the left expression
<i>right</i>	the specified int value in the right expression

### 5.3.2.3 MultExpr() [3/5]

```
MultExpr::MultExpr (
    std::string left,
    int right )
```

Constructs a [MultExpr](#) object with a specified string as the left expression and a specified int value as the right expression.

#### Parameters

<i>left</i>	the specified string in the left expression
<i>right</i>	the specified int value in the right expression

### 5.3.2.4 MultExpr() [4/5]

```
MultExpr::MultExpr (
    int left,
    std::string right )
```

Constructs a [MultExpr](#) object with a specified int as the left expression and a specified string value as the right expression.

#### Parameters

<i>left</i>	the specified int value in the left expression
<i>right</i>	the specified string in the right expression

### 5.3.2.5 MultExpr() [5/5]

```
MultExpr::MultExpr (
    std::string left,
    std::string right )
```

Constructs a [MultExpr](#) object with the specified strings.

#### Parameters

<i>left</i>	the specified string in the left <a href="#">VarExpr</a> expression
<i>right</i>	the specified string in the right <a href="#">VarExpr</a> expression

## 5.3.3 Member Function Documentation

### 5.3.3.1 equals()

```
bool MultExpr::equals (
    Expr * expr ) [virtual]
```

Compared the specified expression with this object for equality.

#### Parameters

<i>expr</i>	the expression to be compared for equality with this object
-------------	---

#### Returns

true if the specified expression is a [MultExpr](#) object and it has the same value with this object. Otherwise returns false.

Implements [Expr](#).

### 5.3.3.2 has\_variable()

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

Check if this [MultExpr](#) expression object contains a [VarExpr](#) object.

#### Returns

true if either its left or right expression contains a [VarExpr](#) object

Implements [Expr](#).

### 5.3.3.3 interp()

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

Multiply the left and right expressions and returns the int value of the result.

#### Returns

the int value of the multiplication of the left and right expressions

Implements [Expr](#).

### 5.3.3.4 pretty\_print()

```
void MultExpr::pretty_print (
    std::ostream & out ) [virtual]
```

Print this object in a prettier way. To specify, we'll omit redundant parentheses and add a space around operators.

#### Parameters

<i>out</i>	the output stream used to print this object
------------	---

Implements [Expr](#).

### 5.3.3.5 print()

```
void MultExpr::print (
    std::ostream & out ) [virtual]
```

Print this object with parentheses and a add sign.

#### Parameters

<i>out</i>	the output stream used to print this object
------------	---

Implements [Expr](#).

### 5.3.3.6 subst()

```
Expr * MultExpr::subst (
    std::string s,
    Expr * expr ) [virtual]
```



Substitute the specified string in this object with a specified expression

**Parameters**

<i>s</i>	the specified string to be substituted
<i>expr</i>	the specified expression used to substitute the string

**Returns**

a new [MultiExpr](#) object with the specified strings in its left and right expressions have been substituted

Implements [Expr](#).

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

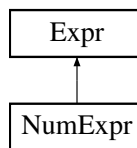
- /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/expr.hpp
- /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/expr.cpp

## 5.4 NumExpr Class Reference

[NumExpr](#) class - for number expressions.

```
#include <expr.hpp>
```

Inheritance diagram for NumExpr:

**Public Member Functions**

- [NumExpr](#) (int v)
- bool [equals](#) ([Expr](#) \*expr)
- int [interp](#) ()
- bool [has\\_variable](#) ()
- [Expr](#) \* [subst](#) (std::string s, [Expr](#) \*expr)
- void [print](#) (std::ostream &out)
- void [pretty\\_print](#) (std::ostream &out)

**Public Member Functions inherited from [Expr](#)**

- virtual bool [equals](#) ([Expr](#) \*expr)=0
- virtual int [interp](#) ()=0
- virtual bool [has\\_variable](#) ()=0
- virtual [Expr](#) \* [subst](#) (std::string s, [Expr](#) \*expr)=0
- virtual void [print](#) (std::ostream &out)=0
- virtual void [pretty\\_print](#) (std::ostream &out)=0
- std::string [to\\_string](#) ()
- precedence\_t [get\\_precedence](#) ()
- std::string [to\\_pretty\\_string](#) ()

## Additional Inherited Members

### Public Attributes inherited from [Expr](#)

- precedence\_t **prec**

### 5.4.1 Detailed Description

[NumExpr](#) class - for number expressions.

### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 NumExpr()

```
NumExpr::NumExpr (
    int v )
```

Constructs a [NumExpr](#) with the specified value and the precedence 0.

#### Parameters

<i>v</i>	the specified value
----------	---------------------

### 5.4.3 Member Function Documentation

#### 5.4.3.1 equals()

```
bool NumExpr::equals (
    Expr * expr ) [virtual]
```

Compared the specified expression with this object for equality.

#### Parameters

<i>expr</i>	the expression to be compared for equality with this object
-------------	---

#### Returns

true if the specified expression is a [NumExpr](#) object and it has the same value with this object. Otherwise returns false.

Implements [Expr](#).

#### 5.4.3.2 has\_variable()

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

Check if this [NumExpr](#) expression object contains a [VarExpr](#) object.

##### Returns

false since a [NumExpr](#) object won't have a variable

Implements [Expr](#).

#### 5.4.3.3 interp()

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

Evaluated the expression and returns the int value.

##### Returns

the int value of the member variable val

Implements [Expr](#).

#### 5.4.3.4 pretty\_print()

```
void NumExpr::pretty_print (
    std::ostream & out ) [virtual]
```

Print this object in a prettier way. To specify, we'll omit redundant parentheses and add space around operators.

##### Parameters

<i>out</i>	the output stream used to print this object
------------	---

Implements [Expr](#).

#### 5.4.3.5 print()

```
void NumExpr::print (
    std::ostream & out ) [virtual]
```

Print this object.

##### Parameters

<i>out</i>	the output stream used to print this object
------------	---

Implements [Expr](#).

#### 5.4.3.6 subst()

```
Expr * NumExpr::subst (
    std::string s,
    Expr * expr ) [virtual]
```

Substitute the specified string in this object with a specified expression

##### Parameters

<i>s</i>	the specified string to be substituted
<i>expr</i>	the specified expression used to substitute the string

##### Returns

a new [NumExpr](#) object with the same value as this object

Implements [Expr](#).

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

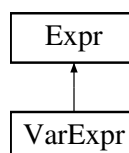
- /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/expr.hpp
- /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/expr.cpp

## 5.5 VarExpr Class Reference

[VarExpr](#) class - for variable expressions.

```
#include <expr.hpp>
```

Inheritance diagram for VarExpr:



## Public Member Functions

- [VarExpr](#) (std::string s)
- bool [equals](#) ([Expr](#) \*expr)
- int [interp](#) ()
- bool [has\\_variable](#) ()
- [Expr](#) \* [subst](#) (std::string s, [Expr](#) \*expr)
- void [print](#) (std::ostream &out)
- void [pretty\\_print](#) (std::ostream &out)

## Public Member Functions inherited from [Expr](#)

- virtual bool [equals](#) ([Expr](#) \*expr)=0
- virtual int [interp](#) ()=0
- virtual bool [has\\_variable](#) ()=0
- virtual [Expr](#) \* [subst](#) (std::string s, [Expr](#) \*expr)=0
- virtual void [print](#) (std::ostream &out)=0
- virtual void [pretty\\_print](#) (std::ostream &out)=0
- std::string [to\\_string](#) ()
- precedence\_t [get\\_precedence](#) ()
- std::string [to\\_pretty\\_string](#) ()

## Additional Inherited Members

### Public Attributes inherited from [Expr](#)

- precedence\_t [prec](#)

## 5.5.1 Detailed Description

[VarExpr](#) class - for variable expressions.

## 5.5.2 Constructor & Destructor Documentation

### 5.5.2.1 [VarExpr](#)()

```
VarExpr::VarExpr (
    std::string s )
```

Constructs a [NumExpr](#) with the specified string and the precedence 0.

#### Parameters

s	the specified string
---	----------------------

### 5.5.3 Member Function Documentation

#### 5.5.3.1 equals()

```
bool VarExpr::equals (
    Expr * expr ) [virtual]
```

Compared the specified expression with this object for equality.

##### Parameters

<i>expr</i>	the expression to be compared for equality with this object
-------------	---

##### Returns

true if the specified expression is a [VarExpr](#) object and it has the same value with this object. Otherwise returns false.

Implements [Expr](#).

#### 5.5.3.2 has\_variable()

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

Check if this [VarExpr](#) expression object contains a [VarExpr](#) object.

##### Returns

true because a [VarExpr](#) object always contains a [VarExpr](#) object

Implements [Expr](#).

#### 5.5.3.3 interp()

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

Returns the int value of this [VarExpr](#) object

##### Returns

throw a runtime error since a varibale cannot be evaluated to a int value

Implements [Expr](#).

#### 5.5.3.4 pretty\_print()

```
void VarExpr::pretty_print (
    std::ostream & out ) [virtual]
```

Print this object in a prettier way.

**Parameters**

<i>out</i>	the output stream used to print this object
------------	---

Implements [Expr](#).

**5.5.3.5 print()**

```
void VarExpr::print (
    std::ostream & out ) [virtual]
```

Print this object.

**Parameters**

<i>out</i>	the output stream used to print this object
------------	---

Implements [Expr](#).

**5.5.3.6 subst()**

```
Expr * VarExpr::subst (
    std::string s,
    Expr * expr ) [virtual]
```

Substitute the specified string in this object with a specified expression

**Parameters**

<i>s</i>	the specified string to be substituted
<i>expr</i>	the specified expression used to substitute the string

**Returns**

the specified expression if s equals val, otherwise return a new [VarExpr](#) object with the same val as this object

Implements [Expr](#).

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

- /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/expr.hpp
- /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/expr.cpp



## Chapter 6

# File Documentation

### 6.1 /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/cmdline.h

```
00001 //
00002 //  cmdline.h
00003 //  hw1
00004 //
00005 //  Created by Yue Sun on 1/12/23.
00006 //
00007
00008 #ifndef cmdline_h
00009 #define cmdline_h
00010
00011 void use_arguments(int argc, const char* argv[]);
00012
00013 #endif /* cmdline_h */
```

### 6.2 /Users/sonia/Documents/MSD/CS6015GitRepo/MSDScript/expr.hpp

```
00001 //
00002 //  expr.hpp
00003 //  HW2_Expression
00004 //
00005 //  Created by Yue Sun on 1/12/23.
00006 //
00007
00008 #ifndef expr_hpp
00009 #define expr_hpp
00010
00011 #include <sstream>
00012 #include <string>
00013
00014 enum precedence_t {
00015     prec_none, // 0
00016     prec_add, // 1
00017     prec_mult, // 2
00018 };
00019
00020 class Expr {
00021 public:
00022     precedence_t prec; // !< the precedence of an expression
00023     virtual bool equals(Expr* expr)=0;
00024     virtual int interp() = 0;
00025     virtual bool has_variable() = 0;
00026     virtual Expr* subst(std::string s, Expr* expr) = 0;
00027     virtual void print(std::ostream& out) = 0;
00028     virtual void pretty_print(std::ostream& out) = 0;
00029     std::string to_string();
00030     precedence_t get_precedence() {
00031         return prec;
00032     }
00033     std::string to_pretty_string();
00034 };
00035
00036 class NumExpr : public Expr {
00037 private:
00038     int val;
```

```

00045 public:
00046     NumExpr(int v);
00047     bool equals(Expr* expr);
00048     int interp();
00049     bool has_variable();
00050     Expr* subst(std::string s, Expr* expr);
00051     void print(std::ostream& out);
00052     void pretty_print(std::ostream& out);
00053 };
00054
00055 class AddExpr : public Expr {
00056 private:
00057     Expr* lhs;
00058     Expr* rhs;
00059 public:
00060     AddExpr(Expr *left, Expr *right);
00061     AddExpr(int left, int right);
00062     AddExpr(std::string left, int right);
00063     AddExpr(int left, std::string right);
00064     AddExpr(std::string left, std::string right);
00065     bool equals(Expr* expr);
00066     int interp();
00067     bool has_variable();
00068     Expr* subst(std::string s, Expr* expr);
00069     void print(std::ostream& out);
00070     void pretty_print(std::ostream& out);
00071 };
00072
00073 class MultExpr : public Expr {
00074 public:
00075     Expr* lhs;
00076     Expr* rhs;
00077 public:
00078     MultExpr(Expr *left, Expr *right);
00079     MultExpr(int left, int right);
00080     MultExpr(std::string left, int right);
00081     MultExpr(int left, std::string right);
00082     MultExpr(std::string left, std::string right);
00083     bool equals(Expr* expr);
00084     int interp();
00085     bool has_variable();
00086     Expr* subst(std::string s, Expr* expr);
00087     void print(std::ostream& out);
00088     void pretty_print(std::ostream& out);
00089 };
00090
00091 class VarExpr : public Expr {
00092 private:
00093     std::string val;
00094 public:
00095     VarExpr(std::string s);
00096     bool equals(Expr* expr);
00097     int interp();
00098     bool has_variable();
00099     Expr* subst(std::string s, Expr* expr);
00100     void print(std::ostream& out);
00101     void pretty_print(std::ostream& out);
00102 };
00103
00104 #endif /* expr_hpp */

```

# Index

AddExpr, [9](#)  
    AddExpr, [10](#), [11](#)  
    equals, [11](#)  
    has\_variable, [12](#)  
    interp, [12](#)  
    pretty\_print, [12](#)  
    print, [13](#)  
    subst, [13](#)

equals  
    AddExpr, [11](#)  
    Expr, [14](#)  
    MultExpr, [19](#)  
    NumExpr, [23](#)  
    VarExpr, [27](#)

Expr, [14](#)  
    equals, [14](#)  
    has\_variable, [14](#)  
    interp, [15](#)  
    pretty\_print, [15](#)  
    print, [15](#)  
    subst, [15](#)  
    to\_pretty\_string, [15](#)  
    to\_string, [16](#)

has\_variable  
    AddExpr, [12](#)  
    Expr, [14](#)  
    MultExpr, [19](#)  
    NumExpr, [24](#)  
    VarExpr, [27](#)

interp  
    AddExpr, [12](#)  
    Expr, [15](#)  
    MultExpr, [19](#)  
    NumExpr, [24](#)  
    VarExpr, [27](#)

MultExpr, [16](#)  
    equals, [19](#)  
    has\_variable, [19](#)  
    interp, [19](#)  
    MultExpr, [17](#), [18](#)  
    pretty\_print, [20](#)  
    print, [20](#)  
    subst, [20](#)

NumExpr, [22](#)  
    equals, [23](#)  
    has\_variable, [24](#)

interp, [24](#)  
    NumExpr, [23](#)  
    pretty\_print, [24](#)  
    print, [24](#)  
    subst, [25](#)

pretty\_print  
    AddExpr, [12](#)  
    Expr, [15](#)  
    MultExpr, [20](#)  
    NumExpr, [24](#)  
    VarExpr, [27](#)

print  
    AddExpr, [13](#)  
    Expr, [15](#)  
    MultExpr, [20](#)  
    NumExpr, [24](#)  
    VarExpr, [28](#)

subst  
    AddExpr, [13](#)  
    Expr, [15](#)  
    MultExpr, [20](#)  
    NumExpr, [25](#)  
    VarExpr, [28](#)

to\_pretty\_string  
    Expr, [15](#)

to\_string  
    Expr, [16](#)

VarExpr, [25](#)  
    equals, [27](#)  
    has\_variable, [27](#)  
    interp, [27](#)  
    pretty\_print, [27](#)  
    print, [28](#)  
    subst, [28](#)  
    VarExpr, [26](#)