

MSDscript

Generated by Doxygen 1.9.6



|  |           |
|--|-----------|
| <b>1 MSDScript</b>                                     | <b>1</b>  |
| <b>2 CS6015</b>  | <b>3</b>  |
| <b>3 Hierarchical Index</b>                            | <b>5</b>  |
| 3.1 Class Hierarchy . . . . .                          | 5         |
| <b>4 Class Index</b>                                   | <b>7</b>  |
| 4.1 Class List . . . . .                               | 7         |
| <b>5 File Index</b>                                    | <b>9</b>  |
| 5.1 File List . . . . .                                | 9         |
| <b>6 Class Documentation</b>                           | <b>11</b> |
| 6.1 Add Class Reference . . . . .                      | 11        |
| 6.1.1 Detailed Description . . . . .                   | 12        |
| 6.1.2 Constructor & Destructor Documentation . . . . . | 12        |
| 6.1.2.1 Add() . . . . .                                | 12        |
| 6.1.3 Member Function Documentation . . . . .          | 13        |
| 6.1.3.1 equals() . . . . .                             | 13        |
| 6.1.3.2 get_prec() . . . . .                           | 13        |
| 6.1.3.3 has_variable() . . . . .                       | 14        |
| 6.1.3.4 interp() . . . . .                             | 14        |
| 6.1.3.5 pretty_print_at() . . . . .                    | 14        |
| 6.1.3.6 print() . . . . .                              | 15        |
| 6.1.3.7 subst() . . . . .                              | 15        |
| 6.2 Expr Class Reference . . . . .                     | 15        |
| 6.2.1 Detailed Description . . . . .                   | 16        |
| 6.2.2 Member Function Documentation . . . . .          | 16        |
| 6.2.2.1 equals() . . . . .                             | 16        |
| 6.2.2.2 get_prec() . . . . .                           | 17        |
| 6.2.2.3 has_variable() . . . . .                       | 17        |
| 6.2.2.4 interp() . . . . .                             | 17        |
| 6.2.2.5 pretty_print() . . . . .                       | 17        |
| 6.2.2.6 pretty_print_at() . . . . .                    | 18        |
| 6.2.2.7 print() . . . . .                              | 18        |
| 6.2.2.8 subst() . . . . .                              | 18        |
| 6.3 Let Class Reference . . . . .                      | 20        |
| 6.3.1 Detailed Description . . . . .                   | 21        |
| 6.3.2 Constructor & Destructor Documentation . . . . . | 21        |
| 6.3.2.1 Let() . . . . .                                | 21        |
| 6.3.3 Member Function Documentation . . . . .          | 22        |
| 6.3.3.1 equals() . . . . .                             | 22        |
| 6.3.3.2 get_prec() . . . . .                           | 22        |

|  |    |
|--|----|
| 6.3.3.3 <code>has_variable()</code> . . . . .          | 23 |
| 6.3.3.4 <code>interp()</code> . . . . .                | 23 |
| 6.3.3.5 <code>pretty_print_at()</code> . . . . .       | 23 |
| 6.3.3.6 <code>print()</code> . . . . .                 | 24 |
| 6.3.3.7 <code>subst()</code> . . . . .                 | 24 |
| 6.4 Mult Class Reference . . . . .                     | 24 |
| 6.4.1 Detailed Description . . . . .                   | 26 |
| 6.4.2 Constructor & Destructor Documentation . . . . . | 26 |
| 6.4.2.1 <code>Mult()</code> . . . . .                  | 26 |
| 6.4.3 Member Function Documentation . . . . .          | 26 |
| 6.4.3.1 <code>equals()</code> . . . . .                | 26 |
| 6.4.3.2 <code>get_prec()</code> . . . . .              | 27 |
| 6.4.3.3 <code>has_variable()</code> . . . . .          | 27 |
| 6.4.3.4 <code>interp()</code> . . . . .                | 27 |
| 6.4.3.5 <code>pretty_print_at()</code> . . . . .       | 27 |
| 6.4.3.6 <code>print()</code> . . . . .                 | 28 |
| 6.4.3.7 <code>subst()</code> . . . . .                 | 28 |
| 6.5 Num Class Reference . . . . .                      | 29 |
| 6.5.1 Detailed Description . . . . .                   | 30 |
| 6.5.2 Constructor & Destructor Documentation . . . . . | 30 |
| 6.5.2.1 <code>Num()</code> . . . . .                   | 30 |
| 6.5.3 Member Function Documentation . . . . .          | 30 |
| 6.5.3.1 <code>equals()</code> . . . . .                | 30 |
| 6.5.3.2 <code>get_prec()</code> . . . . .              | 31 |
| 6.5.3.3 <code>has_variable()</code> . . . . .          | 31 |
| 6.5.3.4 <code>interp()</code> . . . . .                | 31 |
| 6.5.3.5 <code>pretty_print_at()</code> . . . . .       | 31 |
| 6.5.3.6 <code>print()</code> . . . . .                 | 32 |
| 6.5.3.7 <code>subst()</code> . . . . .                 | 32 |
| 6.6 Var Class Reference . . . . .                      | 33 |
| 6.6.1 Detailed Description . . . . .                   | 34 |
| 6.6.2 Constructor & Destructor Documentation . . . . . | 34 |
| 6.6.2.1 <code>Var()</code> . . . . .                   | 34 |
| 6.6.3 Member Function Documentation . . . . .          | 34 |
| 6.6.3.1 <code>equals()</code> . . . . .                | 34 |
| 6.6.3.2 <code>get_prec()</code> . . . . .              | 35 |
| 6.6.3.3 <code>has_variable()</code> . . . . .          | 35 |
| 6.6.3.4 <code>interp()</code> . . . . .                | 35 |
| 6.6.3.5 <code>pretty_print_at()</code> . . . . .       | 35 |
| 6.6.3.6 <code>print()</code> . . . . .                 | 36 |
| 6.6.3.7 <code>subst()</code> . . . . .                 | 36 |

---

|  |           |
|--|-----------|
| <b>7 File Documentation</b>  | <b>37</b> |
| 7.1 /Users/rasonhung/Study/MSD/CS6015/cmdline.h File Reference . . . . . | 37        |
| 7.1.1 Detailed Description . . . . .                                     | 37        |
| 7.1.2 Function Documentation . . . . .                                   | 37        |
| 7.1.2.1 use_arguments() . . . . .  | 37        |
| 7.2 /Users/rasonhung/Study/MSD/CS6015/cmdline.h . . . . .                | 38        |
| 7.3 /Users/rasonhung/Study/MSD/CS6015/expr.h File Reference . . . . .    | 38        |
| 7.3.1 Detailed Description . . . . .                                     | 39        |
| 7.4 /Users/rasonhung/Study/MSD/CS6015/expr.h . . . . .                   | 39        |
| <b>Index</b>   | <b>43</b> |



## Chapter 1

# MSDScript

passing arguments through command line, execute with `-help`, `-test ...`

### Author

Juisheng Hung (Rason)

### Date

01-16-2023





## Chapter 2

## CS6015



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

|                |    |
|----------------|----|
| Expr . . . . . | 15 |
| Add . . . . .  | 11 |
| Let . . . . .  | 20 |
| Mult . . . . . | 24 |
| Num . . . . .  | 29 |
| Var . . . . .  | 33 |



## Chapter 4

# Class Index

### 4.1 Class List

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

|                      |  |                    |
|----------------------|--|--------------------|
| <a href="#">Add</a>  | <a href="#">Add</a> class inherits from <a href="#">Expr</a> class, representing addition for two expressions . . . . .            | <a href="#">11</a> |
| <a href="#">Expr</a> | Abstract expression class<br>(pure abstract class) . . . . .   | <a href="#">15</a> |
| <a href="#">Let</a>  | <a href="#">Let</a> class inherits from <a href="#">Expr</a> class, representing setting values for some expressions if applicable | <a href="#">20</a> |
| <a href="#">Mult</a> | <a href="#">Mult</a> class inherits from <a href="#">Expr</a> class, representing multiplication for two expressions . . . . .     | <a href="#">24</a> |
| <a href="#">Num</a>  | <a href="#">Num</a> class inherits from <a href="#">Expr</a> class, representing pure number . . . . .                             | <a href="#">29</a> |
| <a href="#">Var</a>  | <a href="#">Var</a> class inherits from <a href="#">Expr</a> class, representing pure variable . . . . .                           | <a href="#">33</a> |



## Chapter 5

# File Index

### 5.1 File List

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

|   |    |
|---|----|
| <a href="#">/Users/rasonhung/Study/MSD/CS6015/cmdline.h</a> |    |
| Actual function that executes command line script . . . . . | 37 |
| <a href="#">/Users/rasonhung/Study/MSD/CS6015/expr.h</a>    |    |
| Expression class . . . . .                                  | 38 |





## Chapter 6

# Class Documentation

### 6.1 Add Class Reference

`Add` class inherits from `Expr` class, representing addition for two expressions.

```
#include <expr.h>
```

Inheritance diagram for `Add`:



#### Public Member Functions

- `Add (Expr *lhs, Expr *rhs)`  
*Constructor for `Add` object.*
- `bool equals (Expr *e)` override  
*Judge if this `Add` class object equals to another object.*
- `int interp ()` override  
*Interpret `Add` object to an integer value.*
- `bool has_variable ()` override  
*Judge if the `Add` object contains any `Var`.*
- `Expr * subst (std::string string, Expr *e)` override  
*Substitute the `Var` inside `Add` object with another `Expr`.*
- `void print (std::ostream &ostream)` override  
*print the expression into most basic format (with parentheses, no space)*
- `void pretty_print_at (std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool lastLvlLeft↵ Mult)` override  
*helper function for `pretty_print(std::ostream &ostream)`*
- `precedence_t get_prec ()` override  
*implementation helper function of `pretty_print_at` for classifying case*

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

- virtual bool [equals](#) ([Expr](#) \*e)=0  
*Judge if this [Expr](#) class object equals to another object.*
- virtual int [interp](#) ()=0  
*Interpret [Expr](#) object to an integer value.*
- virtual bool [has\\_variable](#) ()=0  
*Judge if the [Expr](#) object contains any Variable.*
- virtual [Expr](#) \* [subst](#) (std::string string, [Expr](#) \*e)=0  
*Substitute the Variable inside [Expr](#) object with another [Expr](#).*
- virtual void [print](#) (std::ostream &ostream)=0  
*print the expression into most basic format (with parentheses, no space)*
- virtual void [pretty\\_print\\_at](#) (std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool lastLvlMult)=0  
*helper function for [pretty\\_print\(std::ostream &ostream\)](#)*
- virtual precedence\_t [get\\_prec](#) ()=0  
*implementation helper function of [pretty\\_print\\_at](#) for classifying case*
- void [pretty\\_print](#) (std::ostream &ostream)  
*print the expression into a pretty format (avoids unnecessary parentheses, with space around + / \*)*
- std::string [to\\_string](#) ()  
*converting expression to string with basic format*
- std::string [to\\_pretty\\_string](#) ()  
*converting expression to string with a pretty format*

## Public Attributes

- [Expr](#) \* [lhs\\_](#)  
*the [Expr](#) object that makes up the left hand side of the [Add](#) object*
- [Expr](#) \* [rhs\\_](#)  
*the [Expr](#) object that makes up the right hand side of the [Add](#) object*

### 6.1.1 Detailed Description

[Add](#) class inherits from [Expr](#) class, representing addition for two expressions.

### 6.1.2 Constructor & Destructor Documentation

#### 6.1.2.1 Add()

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

Constructor for [Add](#) object.

## Parameters

|            |   |
|------------|---|
| <i>lhs</i> | an <a href="#">Expr</a> object on the left hand side  |
| <i>rhs</i> | an <a href="#">Expr</a> object on the right hand side |

## 6.1.3 Member Function Documentation

### 6.1.3.1 equals()

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

Judge if this [Add](#) class object equals to another object.

## Parameters

|          |  |
|----------|--|
| <i>e</i> | an <a href="#">Expr</a> pointer to <a href="#">Expr</a> object waited to be compared |
|----------|--|

## Returns

returns a boolean, true if two object equals, otherwise false

Implements [Expr](#).

### 6.1.3.2 get\_prec()

```
precedence_t Add::get_prec ( ) [override], [virtual]
```

implementation helper function of pretty\_print\_at for classifying case

## Returns

precedence\_t type enum

Implements [Expr](#).

### 6.1.3.3 has\_variable()

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

Judge if the [Add](#) object contains any [Var](#).

#### Returns

returns a boolean, true if the [Expr](#) object contains any [Var](#), otherwise false

Implements [Expr](#).

### 6.1.3.4 interp()

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

Interpret [Add](#) object to an integer value.

#### Returns

returns the actual integer value (lhs + rhs) of the [Add](#), if it contains [Var](#), throw an exception

Implements [Expr](#).

### 6.1.3.5 pretty\_print\_at()

```
void Add::pretty_print_at (
    std::ostream & ostream,
    std::streampos & lastReturnSeen,
    bool lastLvlLeft,
    bool lastLvlMult ) [override], [virtual]
```

helper function for [pretty\\_print\(std::ostream &ostream\)](#)

#### Parameters

|                       |  |
|-----------------------|--|
| <i>ostream</i>        | deliver string through this output stream  |
| <i>lastReturnSeen</i> | tracking the position of last '\n' seen (generated by <a href="#">Let</a> ) by passing reference                     |
| <i>lastLvlLeft</i>    | tracking where did the last binding came from, return true if it is the left hand side of the upper level expression |
| <i>lastLvlMult</i>    | tracking where did the last binding came from, return true if the upper level expression is a <a href="#">Mult</a>   |

Implements [Expr](#).

### 6.1.3.6 print()

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

print the expression into most basic format (with parentheses, no space)

#### Parameters

|                |   |
|----------------|---|
| <i>ostream</i> | deliver string through this output stream |
|----------------|---|

Implements [Expr](#).

### 6.1.3.7 subst()

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

Substitute the [Var](#) inside [Add](#) object with another [Expr](#).

#### Parameters

|               |  |
|---------------|--|
| <i>string</i> | first argument, a target string that is waited to be substituted   |
| <i>e</i>      | second argument, an <a href="#">Expr</a> pointer to object that is going to substitute the <a href="#">Var</a> inside expression |

#### Returns

returns the new [Expr](#) pointer to object after substitution, return the original object if string variable not found

Implements [Expr](#).

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

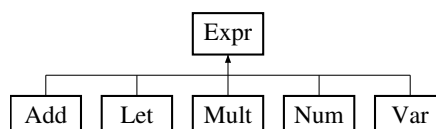
- [/Users/rasonhung/Study/MSD/CS6015/expr.h](#)
- [/Users/rasonhung/Study/MSD/CS6015/expr.cpp](#)

## 6.2 Expr Class Reference

Abstract expression class  
(pure abstract class)

```
#include <expr.h>
```

Inheritance diagram for Expr:



## Public Member Functions

- virtual bool `equals (Expr *e)=0`  
*Judge if this Expr class object equals to another object.*
- virtual int `interp ()=0`  
*Interpret Expr object to an integer value.*
- virtual bool `has_variable ()=0`  
*Judge if the Expr object contains any Variable.*
- virtual `Expr * subst (std::string string, Expr *e)=0`  
*Substitute the Variable inside Expr object with another Expr.*
- virtual void `print (std::ostream &ostream)=0`  
*print the expression into most basic format (with parentheses, no space)*
- virtual void `pretty_print_at (std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvLeft, bool lastLvIMult)=0`  
*helper function for pretty\_print(std::ostream &ostream)*
- virtual precedence\_t `get_prec ()=0`  
*implementation helper function of pretty\_print\_at for classifying case*
- void `pretty_print (std::ostream &ostream)`  
*print the expression into a pretty format (avoids unnecessary parentheses, with space around + / \*)*
- std::string `to_string ()`  
*converting expression to string with basic format*
- std::string `to_pretty_string ()`  
*converting expression to string with a pretty format*

### 6.2.1 Detailed Description

Abstract expression class  
(pure abstract class)

### 6.2.2 Member Function Documentation

#### 6.2.2.1 equals()

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

Judge if this Expr class object equals to another object.

#### Parameters

|                |   |
|----------------|---|
| <code>e</code> | an Expr pointer to object waited to be compared |
|----------------|---|

#### Returns

returns a boolean, true if two object equals, otherwise false

Implemented in [Num](#), [Var](#), [Add](#), [Mult](#), and [Let](#).

### 6.2.2.2 get\_prec()

```
virtual precedence_t Expr::get_prec ( ) [pure virtual]
```

implementation helper function of `pretty_print_at` for classifying case

#### Returns

`precedence_t` type enum

Implemented in [Num](#), [Var](#), [Add](#), [Mult](#), and [Let](#).

### 6.2.2.3 has\_variable()

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

Judge if the [Expr](#) object contains any Variable.

#### Returns

returns a boolean, true if the [Expr](#) object contains any Variable, otherwise false

Implemented in [Num](#), [Var](#), [Add](#), [Mult](#), and [Let](#).

### 6.2.2.4 interp()

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

Interpret [Expr](#) object to an integer value.

#### Returns

returns the actual integer value of the [Expr](#), if it contains Variable, throw an exception

Implemented in [Num](#), [Var](#), [Add](#), [Mult](#), and [Let](#).

### 6.2.2.5 pretty\_print()

```
void Expr::pretty_print (
    std::ostream & ostream )
```

print the expression into a pretty format (avoids unnecessary parentheses, with space around + / \*)

## Parameters

|                |   |
|----------------|---|
| <i>ostream</i> | deliver string through this output stream |
|----------------|---|

**6.2.2.6 pretty\_print\_at()**

```
virtual void Expr::pretty_print_at (
    std::ostream & ostream,
    std::streampos & lastReturnSeen,
    bool lastLvlLeft,
    bool lastLvlMult ) [pure virtual]
```

helper function for [pretty\\_print\(std::ostream &ostream\)](#)

## Parameters

|                       |  |
|-----------------------|--|
| <i>ostream</i>        | deliver string through this output stream  |
| <i>lastReturnSeen</i> | tracking the position of last '\n' seen (generated by <a href="#">Let</a> ) by passing reference                     |
| <i>lastLvlLeft</i>    | tracking where did the last binding came from, return true if it is the left hand side of the upper level expression |
| <i>lastLvlMult</i>    | tracking where did the last binding came from, return true if the upper level expression is a <a href="#">Mult</a>   |

Implemented in [Num](#), [Var](#), [Add](#), [Mult](#), and [Let](#).

**6.2.2.7 print()**

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

print the expression into most basic format (with parentheses, no space)

## Parameters

|                |   |
|----------------|---|
| <i>ostream</i> | deliver string through this output stream |
|----------------|---|

Implemented in [Num](#), [Var](#), [Add](#), [Mult](#), and [Let](#).

**6.2.2.8 subst()**

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



Substitute the Variable inside [Expr](#) object with another [Expr](#).

**Parameters**

|               |   |
|---------------|---|
| <i>string</i> | first argument, a target string that is waited to be substituted  |
| <i>e</i>      | second argument, an <a href="#">Expr</a> pointer to object that is going to substitute the Variable inside expression |

**Returns**

returns the new [Expr](#) pointer to object after substitution, return the original object if string Variable not found

Implemented in [Num](#), [Var](#), [Add](#), [Mult](#), and [Let](#).

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

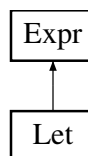
- [/Users/rasonhung/Study/MSD/CS6015/expr.h](#)
- [/Users/rasonhung/Study/MSD/CS6015/expr.cpp](#)

## 6.3 Let Class Reference

[Let](#) class inherits from [Expr](#) class, representing setting values for some expressions if applicable.

```
#include <expr.h>
```

Inheritance diagram for Let:

**Public Member Functions**

- [Let](#) (std::string lhs, [Expr](#) \*rhs, [Expr](#) \*body)  
*Constructor for Let object.*
- bool [equals](#) ([Expr](#) \*e) override  
*Judge if this Let class object equals to another object.*
- int [interp](#) () override  
*Interpret Let object to an integer value.*
- bool [has\\_variable](#) () override  
*Judge if the Let object contains any Variable.*
- [Expr](#) \* [subst](#) (std::string string, [Expr](#) \*e) override  
*Substitute the Var inside Let object with another Expr.*
- void [print](#) (std::ostream &ostream) override  
*print the expression into most basic format (with parentheses, no space)*
- void [pretty\\_print\\_at](#) (std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool lastLvl↵ Mult) override  
*helper function for pretty\_print(std::ostream &ostream)*
- precedence\_t [get\\_prec](#) () override  
*implementation helper function of pretty\_print\_at for classifying case*

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

- virtual bool [equals](#) ([Expr](#) \*e)=0  
*Judge if this [Expr](#) class object equals to another object.*
- virtual int [interp](#) ()=0  
*Interpret [Expr](#) object to an integer value.*
- virtual bool [has\\_variable](#) ()=0  
*Judge if the [Expr](#) object contains any Variable.*
- virtual [Expr](#) \* [subst](#) (std::string string, [Expr](#) \*e)=0  
*Substitute the Variable inside [Expr](#) object with another [Expr](#).*
- virtual void [print](#) (std::ostream &ostream)=0  
*print the expression into most basic format (with parentheses, no space)*
- virtual void [pretty\\_print\\_at](#) (std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool lastLvlMult)=0  
*helper function for [pretty\\_print\(std::ostream &ostream\)](#)*
- virtual precedence\_t [get\\_prec](#) ()=0  
*implementation helper function of [pretty\\_print\\_at](#) for classifying case*
- void [pretty\\_print](#) (std::ostream &ostream)  
*print the expression into a pretty format (avoids unnecessary parentheses, with space around + / \*)*
- std::string [to\\_string](#) ()  
*converting expression to string with basic format*
- std::string [to\\_pretty\\_string](#) ()  
*converting expression to string with a pretty format*

### Public Attributes

- std::string [lhs\\_](#)  
*the expression that is waiting to be set with value*
- [Expr](#) \* [rhs\\_](#)  
*the setting value*
- [Expr](#) \* [body\\_](#)  
*in which expression the variable is set with the value*

#### 6.3.1 Detailed Description

[Let](#) class inherits from [Expr](#) class, representing setting values for some expressions if applicable.

#### 6.3.2 Constructor & Destructor Documentation

##### 6.3.2.1 Let()

```
Let::Let (
    std::string lhs,
    Expr * rhs,
    Expr * body )
```

Constructor for [Let](#) object.

**Parameters**

|             |   |
|-------------|---|
| <i>lhs</i>  | string that represents the variable waiting to be set                 |
| <i>rhs</i>  | an <a href="#">Expr</a> with some value passing to the lhs expression |
| <i>body</i> | in which expression the variable is set with the value                |

**6.3.3 Member Function Documentation****6.3.3.1 equals()**

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

Judge if this [Let](#) class object equals to another object.

**Parameters**

|          |  |
|----------|--|
| <i>e</i> | an <a href="#">Expr</a> pointer to <a href="#">Expr</a> object waited to be compared |
|----------|--|

**Returns**

returns a boolean, true if two object equals, otherwise false

Implements [Expr](#).

**6.3.3.2 get\_prec()**

```
precedence_t Let::get_prec ( )  [override], [virtual]
```

implementation helper function of pretty\_print\_at for classifying case

**Returns**

precedence\_t type enum

Implements [Expr](#).

### 6.3.3.3 has\_variable()

```
bool Let::has_variable ( ) [override], [virtual]
```

Judge if the [Let](#) object contains any Variable.

#### Returns

returns a boolean, always return false

Implements [Expr](#).

### 6.3.3.4 interp()

```
int Let::interp ( ) [override], [virtual]
```

Interpret [Let](#) object to an integer value.

#### Returns

returns the actual integer value of the [Num](#)

Implements [Expr](#).

### 6.3.3.5 pretty\_print\_at()

```
void Let::pretty_print_at (
    std::ostream & ostream,
    std::streampos & lastReturnSeen,
    bool lastLvlLeft,
    bool lastLvlMult ) [override], [virtual]
```

helper function for [pretty\\_print\(std::ostream &ostream\)](#)

#### Parameters

|                       |  |
|-----------------------|--|
| <i>ostream</i>        | deliver string through this output stream  |
| <i>lastReturnSeen</i> | tracking the position of last '\n' seen (generated by <a href="#">Let</a> ) by passing reference                     |
| <i>lastLvlLeft</i>    | tracking where did the last binding came from, return true if it is the left hand side of the upper level expression |
| <i>lastLvlMult</i>    | tracking where did the last binding came from, return true if the upper level expression is a <a href="#">Mult</a>   |

Implements [Expr](#).

### 6.3.3.6 print()

```
void Let::print (
    std::ostream & ostream ) [override], [virtual]
```

print the expression into most basic format (with parentheses, no space)

#### Parameters

|                |   |
|----------------|---|
| <i>ostream</i> | deliver string through this output stream |
|----------------|---|

Implements [Expr](#).

### 6.3.3.7 subst()

```
Expr * Let::subst (
    std::string string,
    Expr * e ) [override], [virtual]
```

Substitute the [Var](#) inside [Let](#) object with another [Expr](#).

#### Parameters

|               |  |
|---------------|--|
| <i>string</i> | first argument, a target string that is waited to be substituted   |
| <i>e</i>      | second argument, an <a href="#">Expr</a> pointer to object that is going to substitute the <a href="#">Var</a> inside expression |

#### Returns

returns this object, since there is no [Var](#) in [Let](#) object

Implements [Expr](#).

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

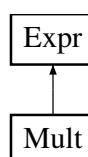
- /Users/rasonhung/Study/MSD/CS6015/[expr.h](#)
- /Users/rasonhung/Study/MSD/CS6015/[expr.cpp](#)

## 6.4 Mult Class Reference

[Mult](#) class inherits from [Expr](#) class, representing multiplication for two expressions.

```
#include <expr.h>
```

Inheritance diagram for Mult:



## Public Member Functions

- **Mult** (**Expr** \*lhs, **Expr** \*rhs)  
*Constructor for **Mult** object.*
- bool **equals** (**Expr** \*e) override  
*Judge if this **Mult** class object equals to another object.*
- int **interp** () override  
*Interpret **Mult** object to an integer value.*
- bool **has\_variable** () override  
*Judge if the **Mult** object contains any **Var**.*
- **Expr** \* **subst** (std::string string, **Expr** \*e) override  
*Substitute the **Var** inside **Mult** object with another **Expr**.*
- void **print** (std::ostream &ostream) override  
*print the expression into most basic format (with parentheses, no space)*
- void **pretty\_print\_at** (std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool lastLvl↵  
Mult) override  
*helper function for **pretty\_print**(std::ostream &ostream)*
- precedence\_t **get\_prec** () override  
*implementation helper function of **pretty\_print\_at** for classifying case*

## Public Member Functions inherited from **Expr**

- virtual bool **equals** (**Expr** \*e)=0  
*Judge if this **Expr** class object equals to another object.*
- virtual int **interp** ()=0  
*Interpret **Expr** object to an integer value.*
- virtual bool **has\_variable** ()=0  
*Judge if the **Expr** object contains any Variable.*
- virtual **Expr** \* **subst** (std::string string, **Expr** \*e)=0  
*Substitute the Variable inside **Expr** object with another **Expr**.*
- virtual void **print** (std::ostream &ostream)=0  
*print the expression into most basic format (with parentheses, no space)*
- virtual void **pretty\_print\_at** (std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool lastLvlMult)=0  
*helper function for **pretty\_print**(std::ostream &ostream)*
- virtual precedence\_t **get\_prec** ()=0  
*implementation helper function of **pretty\_print\_at** for classifying case*
- void **pretty\_print** (std::ostream &ostream)  
*print the expression into a pretty format (avoids unnecessary parentheses, with space around + / \*)*
- std::string **to\_string** ()  
*converting expression to string with basic format*
- std::string **to\_pretty\_string** ()  
*converting expression to string with a pretty format*

## Public Attributes

- **Expr** \* **lhs\_**  
*the **Expr** object that makes up the left hand side of the **Mult** object*
- **Expr** \* **rhs\_**  
*the **Expr** object that makes up the right hand side of the **Mult** object*

### 6.4.1 Detailed Description

[Mult](#) class inherits from [Expr](#) class, representing multiplication for two expressions.

### 6.4.2 Constructor & Destructor Documentation

#### 6.4.2.1 Mult()

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

Constructor for [Mult](#) object.

##### Parameters

|            |   |
|------------|---|
| <i>lhs</i> | an <a href="#">Expr</a> object on the left hand side  |
| <i>rhs</i> | an <a href="#">Expr</a> object on the right hand side |

### 6.4.3 Member Function Documentation

#### 6.4.3.1 equals()

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

Judge if this [Mult](#) class object equals to another object.

##### Parameters

|          |  |
|----------|--|
| <i>e</i> | an <a href="#">Expr</a> pointer to <a href="#">Expr</a> object waited to be compared |
|----------|--|

##### Returns

returns a boolean, true if two object equals, otherwise false

Implements [Expr](#).



#### 6.4.3.2 get\_prec()

```
precedence_t Mult::get_prec ( ) [override], [virtual]
```

implementation helper function of `pretty_print_at` for classifying case

##### Returns

`precedence_t` type enum

Implements [Expr](#).

#### 6.4.3.3 has\_variable()

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

Judge if the [Mult](#) object contains any [Var](#).

##### Returns

returns a boolean, true if the [Expr](#) object contains any [Var](#), otherwise false

Implements [Expr](#).

#### 6.4.3.4 interp()

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

Interpret [Mult](#) object to an integer value.

##### Returns

returns the actual integer value (`lhs * rhs`) of the [Mult](#), if it contains [Var](#), throw an exception

Implements [Expr](#).

#### 6.4.3.5 pretty\_print\_at()

```
void Mult::pretty_print_at (
    std::ostream & ostream,
    std::streampos & lastReturnSeen,
    bool lastLvlLeft,
    bool lastLvlMult ) [override], [virtual]
```

helper function for `pretty_print(std::ostream &ostream)`

## Parameters

|                       |  |
|-----------------------|--|
| <i>ostream</i>        | deliver string through this output stream  |
| <i>lastReturnSeen</i> | tracking the position of last '\n' seen (generated by <a href="#">Let</a> ) by passing reference                     |
| <i>lastLvlLeft</i>    | tracking where did the last binding came from, return true if it is the left hand side of the upper level expression |
| <i>lastLvlMult</i>    | tracking where did the last binding came from, return true if the upper level expression is a <a href="#">Mult</a>   |

Implements [Expr](#).

### 6.4.3.6 print()

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

print the expression into most basic format (with parentheses, no space)

## Parameters

|                |   |
|----------------|---|
| <i>ostream</i> | deliver string through this output stream |
|----------------|---|

Implements [Expr](#).

### 6.4.3.7 subst()

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

Substitute the [Var](#) inside [Mult](#) object with another [Expr](#).

## Parameters

|               |  |
|---------------|--|
| <i>string</i> | first argument, a target string that is waited to be substituted   |
| <i>e</i>      | second argument, an <a href="#">Expr</a> pointer to object that is going to substitute the <a href="#">Var</a> inside expression |

## Returns

returns the new [Expr](#) pointer to object after substitution, return the original object if string variable not found

Implements [Expr](#).

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

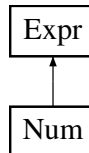
- [/Users/rasonhung/Study/MSD/CS6015/expr.h](#)
- [/Users/rasonhung/Study/MSD/CS6015/expr.cpp](#)

## 6.5 Num Class Reference

`Num` class inherits from `Expr` class, representing pure number.

```
#include <expr.h>
```

Inheritance diagram for `Num`:



### Public Member Functions

- `Num` (int val)  
*Constructor for `Num` object.*
- bool `equals` (`Expr` \*e) override  
*Judge if this `Num` class object equals to another object.*
- int `interp` () override  
*Interpret `Num` object to an integer value.*
- bool `has_variable` () override  
*Judge if the `Num` object contains any `Variable`.*
- `Expr` \* `subst` (std::string string, `Expr` \*e) override  
*Substitute the `Variable` inside `Num` object with another `Expr`.*
- void `print` (std::ostream &ostream) override  
*print the expression into most basic format (with parentheses, no space)*
- void `pretty_print_at` (std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool lastLvlMult) override  
*helper function for `pretty_print(std::ostream &ostream)`*
- precedence\_t `get_prec` () override  
*implementation helper function of `pretty_print_at` for classifying case*

### Public Member Functions inherited from `Expr`

- virtual bool `equals` (`Expr` \*e)=0  
*Judge if this `Expr` class object equals to another object.*
- virtual int `interp` ()=0  
*Interpret `Expr` object to an integer value.*
- virtual bool `has_variable` ()=0  
*Judge if the `Expr` object contains any `Variable`.*
- virtual `Expr` \* `subst` (std::string string, `Expr` \*e)=0  
*Substitute the `Variable` inside `Expr` object with another `Expr`.*
- virtual void `print` (std::ostream &ostream)=0  
*print the expression into most basic format (with parentheses, no space)*
- virtual void `pretty_print_at` (std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool lastLvlMult)=0  
*helper function for `pretty_print(std::ostream &ostream)`*
- virtual precedence\_t `get_prec` ()=0

- implementation helper function of pretty\_print\_at for classifying case*
- void `pretty_print` (std::ostream &ostream)  
*print the expression into a pretty format (avoids unnecessary parentheses, with space around + / \*)*
- std::string `to_string` ()  
*converting expression to string with basic format*
- std::string `to_pretty_string` ()  
*converting expression to string with a pretty format*

## Public Attributes

- int `val_`  
*the integer value of the [Num](#) object*

## 6.5.1 Detailed Description

[Num](#) class inherits from [Expr](#) class, representing pure number.

## 6.5.2 Constructor & Destructor Documentation

### 6.5.2.1 Num()

```
Num::Num (
    int val ) [explicit]
```

Constructor for [Num](#) object.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>val</code> | integer value of <a href="#">Num</a> |
|------------------|--------------------------------------|

## 6.5.3 Member Function Documentation

### 6.5.3.1 equals()

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

Judge if this [Num](#) class object equals to another object.

## Parameters

|                |  |
|----------------|--|
| <code>e</code> | an <a href="#">Expr</a> pointer to <a href="#">Expr</a> object waited to be compared |
|----------------|--|

## Returns

returns a boolean, true if two object equals, otherwise false

Implements [Expr](#).

**6.5.3.2 get\_prec()**

```
precedence_t Num::get_prec ( ) [override], [virtual]
```

implementation helper function of `pretty_print_at` for classifying case

## Returns

`precedence_t` type enum

Implements [Expr](#).

**6.5.3.3 has\_variable()**

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

Judge if the [Num](#) object contains any [Variable](#).

## Returns

returns a boolean, always return false

Implements [Expr](#).

**6.5.3.4 interp()**

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

Interpret [Num](#) object to an integer value.

## Returns

returns the actual integer value of the [Num](#)

Implements [Expr](#).

**6.5.3.5 pretty\_print\_at()**

```
void Num::pretty_print_at (
    std::ostream & ostream,
    std::streampos & lastReturnSeen,
    bool lastLvlLeft,
    bool lastLvlMult ) [override], [virtual]
```

helper function for `pretty_print(std::ostream &ostream)`

## Parameters

|                       |  |
|-----------------------|--|
| <i>ostream</i>        | deliver string through this output stream  |
| <i>lastReturnSeen</i> | tracking the position of last '\n' seen (generated by <a href="#">Let</a> ) by passing reference                     |
| <i>lastLvlLeft</i>    | tracking where did the last binding came from, return true if it is the left hand side of the upper level expression |
| <i>lastLvlMult</i>    | tracking where did the last binding came from, return true if the upper level expression is a <a href="#">Mult</a>   |

Implements [Expr](#).

### 6.5.3.6 print()

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

print the expression into most basic format (with parentheses, no space)

## Parameters

|                |   |
|----------------|---|
| <i>ostream</i> | deliver string through this output stream |
|----------------|---|

Implements [Expr](#).

### 6.5.3.7 subst()

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

Substitute the Variable inside [Num](#) object with another [Expr](#).

## Parameters

|               |   |
|---------------|---|
| <i>string</i> | first argument, a target string that is waited to be substituted  |
| <i>e</i>      | second argument, an <a href="#">Expr</a> pointer to object that is going to substitute the Variable inside expression |

## Returns

returns this object, since there is no Variable in [Num](#) object

Implements [Expr](#).

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

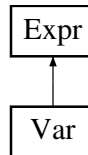
- [/Users/rasonhung/Study/MSD/CS6015/expr.h](#)
- [/Users/rasonhung/Study/MSD/CS6015/expr.cpp](#)

## 6.6 Var Class Reference

`Var` class inherits from `Expr` class, representing pure variable.

```
#include <expr.h>
```

Inheritance diagram for `Var`:



### Public Member Functions

- `Var (std::string varName)`  
*Constructor for `Var` object.*
- `bool equals (Expr *e) override`  
*Judge if this `Var` class object equals to another object, overrides function in superclass.*
- `int interp () override`  
*Interpret `Var` object to an integer value.*
- `bool has_variable () override`  
*Judge if the `Var` object contains any `Var`.*
- `Expr * subst (std::string string, Expr *e) override`  
*Substitute the `Var` object with another `Expr`.*
- `void print (std::ostream &ostream) override`  
*print the expression into most basic format (with parentheses, no space)*
- `void pretty_print_at (std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool lastLvlMult) override`  
*helper function for `pretty_print(std::ostream &ostream)`*
- `precedence_t get_prec () override`  
*implementation helper function of `pretty_print_at` for classifying case*

### Public Member Functions inherited from `Expr`

- `virtual bool equals (Expr *e)=0`  
*Judge if this `Expr` class object equals to another object.*
- `virtual int interp ()=0`  
*Interpret `Expr` object to an integer value.*
- `virtual bool has_variable ()=0`  
*Judge if the `Expr` object contains any Variable.*
- `virtual Expr * subst (std::string string, Expr *e)=0`  
*Substitute the Variable inside `Expr` object with another `Expr`.*
- `virtual void print (std::ostream &ostream)=0`  
*print the expression into most basic format (with parentheses, no space)*
- `virtual void pretty_print_at (std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool lastLvlMult)=0`  
*helper function for `pretty_print(std::ostream &ostream)`*
- `virtual precedence_t get_prec ()=0`

- implementation helper function of pretty\_print\_at for classifying case*
  - void `pretty_print` (std::ostream &ostream)
   
*print the expression into a pretty format (avoids unnecessary parentheses, with space around + / \*)*
  - std::string `to_string` ()
   
*converting expression to string with basic format*
  - std::string `to_pretty_string` ()
   
*converting expression to string with a pretty format*

## Public Attributes

- std::string `string_`
  
*the string name that makes up the `Var` object*

### 6.6.1 Detailed Description

`Var` class inherits from `Expr` class, representing pure variable.

### 6.6.2 Constructor & Destructor Documentation

#### 6.6.2.1 Var()

```
Var::Var (
    std::string varName ) [explicit]
```

Constructor for `Var` object.

#### Parameters

|                      |  |
|----------------------|--|
| <code>varName</code> | a string that can be seen as the label of the <code>Var</code> |
|----------------------|--|

### 6.6.3 Member Function Documentation

#### 6.6.3.1 equals()

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

Judge if this `Var` class object equals to another object, overrides function in superclass.



## Parameters

|                |  |
|----------------|--|
| <code>e</code> | an <a href="#">Expr</a> pointer to <a href="#">Expr</a> object waited to be compared |
|----------------|--|

## Returns

returns a boolean, true if two object equals, otherwise false

Implements [Expr](#).

**6.6.3.2 get\_prec()**

```
precedence_t Var::get_prec ( ) [override], [virtual]
```

implementation helper function of `pretty_print_at` for classifying case

## Returns

`precedence_t` type enum

Implements [Expr](#).

**6.6.3.3 has\_variable()**

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

Judge if the [Var](#) object contains any [Var](#).

## Returns

returns a boolean, always return true

Implements [Expr](#).

**6.6.3.4 interp()**

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

Interpret [Var](#) object to an integer value.

## Returns

A [Var](#) doesn't have specific integer value, throw an exception

Implements [Expr](#).

**6.6.3.5 pretty\_print\_at()**

```
void Var::pretty_print_at (
    std::ostream & ostream,
    std::streampos & lastReturnSeen,
    bool lastLvlLeft,
    bool lastLvlMult ) [override], [virtual]
```

helper function for `pretty_print(std::ostream &ostream)`

## Parameters

|                       |  |
|-----------------------|--|
| <i>ostream</i>        | deliver string through this output stream  |
| <i>lastReturnSeen</i> | tracking the position of last '\n' seen (generated by <a href="#">Let</a> ) by passing reference                     |
| <i>lastLvlLeft</i>    | tracking where did the last binding came from, return true if it is the left hand side of the upper level expression |
| <i>lastLvlMult</i>    | tracking where did the last binding came from, return true if the upper level expression is a <a href="#">Mult</a>   |

Implements [Expr](#).

### 6.6.3.6 print()

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

print the expression into most basic format (with parentheses, no space)

## Parameters

|                |   |
|----------------|---|
| <i>ostream</i> | deliver string through this output stream |
|----------------|---|

Implements [Expr](#).

### 6.6.3.7 subst()

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

Substitute the [Var](#) object with another [Expr](#).

## Parameters

|               |  |
|---------------|--|
| <i>string</i> | first argument, a target string that is waited to be substituted   |
| <i>e</i>      | second argument, an <a href="#">Expr</a> pointer to object that is going to substitute the <a href="#">Var</a> inside expression |

## Returns

returns the new [Expr](#) pointer to object after substitution, return the original object if string variable not found

Implements [Expr](#).

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

- /Users/rasonhung/Study/MSD/CS6015/[expr.h](#)
- /Users/rasonhung/Study/MSD/CS6015/[expr.cpp](#)

## Chapter 7

# File Documentation

### 7.1 /Users/rasonhung/Study/MSD/CS6015/cmdline.h File Reference

actual function that executes command line script

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

#### Functions

- void [use\\_arguments](#) (int argc, const char \*argv[])  
*Take arguments from command line as input, execute corresponding output as required.*

#### 7.1.1 Detailed Description

actual function that executes command line script

#### 7.1.2 Function Documentation

##### 7.1.2.1 use\_arguments()

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

Take arguments from command line as input, execute corresponding output as required.

## Parameters

|             |  |
|-------------|--|
| <i>argc</i> | first argument, the integer number of arguments passed into                              |
| <i>argv</i> | second argument, the pointer to the array of characters that is passed into as parameter |

## Returns

returns void

"--help": if it is the next argument after program name, print out help message, and do not examine other arguments

"--test": if it is the only argument after program name, then print out test result, otherwise, will be treated as invalid argument input any other strings as input: invalid argument, exit the program with 1

## 7.2 /Users/rasonhung/Study/MSD/CS6015/cmdline.h

[Go to the documentation of this file.](#)

```
00001 //
00002 //  cmdline.h
00003 //  CommandLine
00004 //
00005 //  Created by Rason Hung on 1/16/23.
00006 //
00007
00014 #pragma include once
00015 #include <iostream>
00016 #include <string>
00017
00025 void use_arguments(int argc, const char *argv[]);
```

## 7.3 /Users/rasonhung/Study/MSD/CS6015/expr.h File Reference

expression class

```
#include <cstdio>
#include <string>
#include <sstream>
#include <stdexcept>
#include <utility>
```

### Classes

- class [Expr](#)  
*Abstract expression class*  
*(pure abstract class)*
- class [Num](#)  
*Num class inherits from Expr class, representing pure number.*
- class [Var](#)  
*Var class inherits from Expr class, representing pure variable.*
- class [Add](#)  
*Add class inherits from Expr class, representing addition for two expressions.*
- class [Mult](#)  
*Mult class inherits from Expr class, representing multiplication for two expressions.*
- class [Let](#)  
*Let class inherits from Expr class, representing setting values for some expressions if applicable.*

## Enumerations

- enum `precedence_t` { `prec_none` , `prec_add` , `prec_mult` , `prec_let` }

### 7.3.1 Detailed Description

expression class

Contains the blueprint of the superclass - [Expr](#), with its subclass - [Num](#), [Add](#), [Mult](#), [Variable](#)

## 7.4 /Users/rasonhung/Study/MSD/CS6015/expr.h

[Go to the documentation of this file.](#)

```

00001 //
00002 //  expr.h
00003 //  ExpressionClasses
00004 //
00005 //  Created by Rason Hung on 1/22/23.
00006 //
00007
00015 #pragma include once
00016 #include <cstdio>
00017 #include <string>
00018 #include <sstream>
00019 #include <stdexcept>
00020 #include <utility>
00021
00022 typedef enum {
00023     prec_none,    // = 0
00024     prec_add,     // = 1
00025     prec_mult,    // = 2
00026     prec_let,     // = 3
00027 } precedence_t;
00028
00029
00033 class Expr {
00034 public:
00040     virtual bool equals(Expr *e) = 0;
00041
00046     virtual int interp() = 0;
00047
00052     virtual bool has_variable() = 0;
00053
00060     virtual Expr* subst(std::string string, Expr* e)=0;
00061
00062     //TODO: do we need to handle with negative expression?
00067     virtual void print(std::ostream &ostream) = 0;
00068
00076     virtual void pretty_print_at(std::ostream &ostream, std::streampos &lastReturnSeen, bool
lastLvlLeft, bool lastLvlMult) = 0;
00077
00082     virtual precedence_t get_prec() = 0;
00083
00088     void pretty_print(std::ostream &ostream);
00089
00093     std::string to_string();
00094
00098     std::string to_pretty_string(); // if not required - only for test use
00099
00100     //judge if at least with someone's lhs
00101     //    bool isOnAnyLhs(bool &onLhs);
00102 };
00103
00104
00105
00106
00109 class Num : public Expr {
00110 public:
00111     int val_;
00112
00117     explicit Num(int val);
00118
00124     bool equals(Expr *e) override;
00125

```

```

00130     int interp() override;
00131
00136     bool has_variable() override;
00137
00144     Expr* subst(std::string string, Expr* e) override;
00145
00146     void print(std::ostream &ostream) override;
00147
00148     void pretty_print_at(std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool
lastLvlMult) override;
00149
00150     precedence_t get_prec() override;
00151 };
00152
00153
00154
00157 class Var : public Expr {
00158 public:
00159     std::string string_;
00160
00165     explicit Var(std::string varName);
00166
00172     bool equals(Expr *e) override;
00173
00178     int interp() override;
00179
00184     bool has_variable() override;
00185
00192     Expr* subst(std::string string, Expr* e) override;
00193
00194     void print(std::ostream &ostream) override;
00195
00196     void pretty_print_at(std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool
lastLvlMult) override;
00197
00198     precedence_t get_prec() override;
00199 };
00200
00201
00202
00203
00206 class Add : public Expr {
00207 public:
00208     Expr *lhs_;
00209     Expr *rhs_;
00210
00216     Add(Expr *lhs, Expr *rhs);
00217
00223     bool equals(Expr *e) override;
00224
00229     int interp() override;
00230
00235     bool has_variable() override;
00236
00243     Expr* subst(std::string string, Expr* e) override;
00244
00245
00246     void print(std::ostream &ostream) override;
00247
00248     void pretty_print_at(std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool
lastLvlMult) override;
00249
00250     precedence_t get_prec() override;
00251 };
00252
00253
00254
00255
00258 class Mult : public Expr {
00259 public:
00260     Expr *lhs_;
00261     Expr *rhs_;
00262
00268     Mult(Expr *lhs, Expr *rhs);
00269
00275     bool equals(Expr *e) override;
00276
00281     int interp() override;
00282
00287     bool has_variable() override;
00288
00295     Expr* subst(std::string string, Expr* e) override;
00296
00297     void print(std::ostream &ostream) override;
00298
00299     void pretty_print_at(std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool
lastLvlMult) override;

```

```
00300
00301     precedence_t get_prec() override;
00302 };
00303
00306 class Let : public Expr {
00307 public:
00308     std::string lhs_;
00309     Expr *rhs_;
00310     Expr *body_;
00311
00318     Let(std::string lhs, Expr* rhs, Expr* body);
00319
00325     bool equals(Expr *e) override;
00326
00331     int interp() override;
00332
00337     bool has_variable() override;
00338
00345     Expr* subst(std::string string, Expr* e) override;
00346
00347     void print(std::ostream &ostream) override;
00348
00349     void pretty_print_at(std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool
lastLvlMult) override;
00350
00351     precedence_t get_prec() override;
00352 };
```





# Index

/Users/rasonhung/Study/MSD/CS6015/cmdline.h, [37](#)

/Users/rasonhung/Study/MSD/CS6015/expr.h, [38](#)

Add, [11](#)

Add, [12](#)

equals, [13](#)

get\_prec, [13](#)

has\_variable, [13](#)

interp, [14](#)

pretty\_print\_at, [14](#)

print, [14](#)

subst, [15](#)

cmdline.h

use\_arguments, [37](#)

equals

Add, [13](#)

Expr, [16](#)

Let, [22](#)

Mult, [26](#)

Num, [30](#)

Var, [34](#)

Expr, [15](#)

equals, [16](#)

get\_prec, [17](#)

has\_variable, [17](#)

interp, [17](#)

pretty\_print, [17](#)

pretty\_print\_at, [18](#)

print, [18](#)

subst, [18](#)

get\_prec

Add, [13](#)

Expr, [17](#)

Let, [22](#)

Mult, [26](#)

Num, [31](#)

Var, [35](#)

has\_variable

Add, [13](#)

Expr, [17](#)

Let, [22](#)

Mult, [27](#)

Num, [31](#)

Var, [35](#)

interp

Add, [14](#)

Expr, [17](#)

Let, [23](#)

Mult, [27](#)

Num, [31](#)

Var, [35](#)

Let, [20](#)

equals, [22](#)

get\_prec, [22](#)

has\_variable, [22](#)

interp, [23](#)

Let, [21](#)

pretty\_print\_at, [23](#)

print, [23](#)

subst, [24](#)

Mult, [24](#)

equals, [26](#)

get\_prec, [26](#)

has\_variable, [27](#)

interp, [27](#)

Mult, [26](#)

pretty\_print\_at, [27](#)

print, [28](#)

subst, [28](#)

Num, [29](#)

equals, [30](#)

get\_prec, [31](#)

has\_variable, [31](#)

interp, [31](#)

Num, [30](#)

pretty\_print\_at, [31](#)

print, [32](#)

subst, [32](#)

pretty\_print

Expr, [17](#)

pretty\_print\_at

Add, [14](#)

Expr, [18](#)

Let, [23](#)

Mult, [27](#)

Num, [31](#)

Var, [35](#)

print

Add, [14](#)

Expr, [18](#)

Let, [23](#)

Mult, [28](#)

Num, [32](#)  
Var, [36](#)

subst

Add, [15](#)  
Expr, [18](#)  
Let, [24](#)  
Mult, [28](#)  
Num, [32](#)  
Var, [36](#)

use\_arguments

cmdline.h, [37](#)

Var, [33](#)

equals, [34](#)  
get\_prec, [35](#)  
has\_variable, [35](#)  
interp, [35](#)  
pretty\_print\_at, [35](#)  
print, [36](#)  
subst, [36](#)  
Var, [34](#)