MSDscript

Generated by Doxygen 1.9.6

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:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

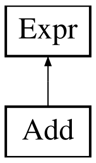
Class Documentation

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 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 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

lhs	an Expr object on the left hand side
rhs	an Expr 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

e	an Expr pointer to Expr 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

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

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

ostream	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

string	first argument, a target string that is waited to be substituted
e	second argument, an Expr pointer to object that is going to substitute the Var 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:

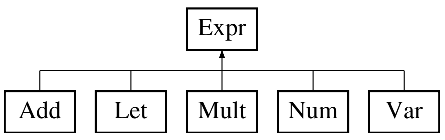
- /Users/rasonhung/Study/MSD/CS6015/expr.h
- /Users/rasonhung/Study/MSD/CS6015/expr.cpp

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 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

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

e	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

ostream	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

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

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

ostream	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

string	first argument, a target string that is waited to be substituted
e	second argument, an Expr 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:

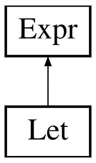
- /Users/rasonhung/Study/MSD/CS6015/expr.h
- /Users/rasonhung/Study/MSD/CS6015/expr.cpp

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

lhs	string that represents the variable waiting to be set
rhs	an Expr with some value passing to the lhs expression
body	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

e	an Expr pointer to Expr 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

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

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

ostream	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

string	first argument, a target string that is waited to be substituted
e	second argument, an Expr pointer to object that is going to substitute the Var 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:

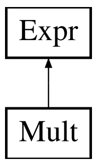
- /Users/rasonhung/Study/MSD/CS6015/expr.h
- /Users/rasonhung/Study/MSD/CS6015/expr.cpp

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

lhs	an Expr object on the left hand side
rhs	an Expr 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

e	an Expr pointer to Expr 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

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

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

ostream	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

string	first argument, a target string that is waited to be substituted
e	second argument, an Expr pointer to object that is going to substitute the Var 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:

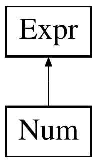
- /Users/rasonhung/Study/MSD/CS6015/expr.h
- /Users/rasonhung/Study/MSD/CS6015/expr.cpp

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 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

- 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

val	integer value of Num

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

e	an Expr pointer to Expr 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

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

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

ostream	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

string	first argument, a target string that is waited to be substituted
e	second argument, an Expr 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:

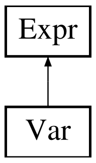
- /Users/rasonhung/Study/MSD/CS6015/expr.h
- /Users/rasonhung/Study/MSD/CS6015/expr.cpp

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 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 **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

varName	a string that can be seen as the label of the Var

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

e	an Expr pointer to Expr 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

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

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

ostream	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

string	first argument, a target string that is waited to be substituted
e	second argument, an Expr pointer to object that is going to substitute the Var 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

argc	first argument, the integer number of arguments passed into
argv	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.

```
//
//  cmdline.h
//  CommandLine
//
//  Created by Rason Hung on 1/16/23.
//

#pragma include once
#include <iostream>
#include <string>

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.

```
//
//  expr.h
//  ExpressionClasses
//
//  Created by Rason Hung on 1/22/23.
//

#pragma once
#include <cstdio>
#include <string>
#include <sstream>
#include <stdexcept>
#include <utility>

typedef enum {
    prec_none,  // = 0
    prec_add,   // = 1
    prec_mult,  // = 2
    prec_let,   // = 3
} precedence_t;


class Expr {
public:
    virtual bool equals(Expr *e) = 0;

    virtual int interp() = 0;

    virtual bool has_variable() = 0;

    virtual Expr* subst(std::string string, Expr* e)=0;

    //TODO: do we need to handle with negative expression?
    virtual void print(std::ostream &ostream) = 0;

    virtual void pretty_print_at(std::ostream &ostream, std::streampos &lastReturnSeen, bool
lastLvlLeft, bool lastLvlMult) = 0;

    virtual precedence_t get_prec() = 0;

    void pretty_print(std::ostream &ostream);

    std::string to_string();

    std::string to_pretty_string(); // if not required - only for test use

    //judge if at least with someone's lhs
//    bool isOnAnyLhs(bool &onLhs);
};




class Num : public Expr {
public:
    int val_;

    explicit Num(int val);

    bool equals(Expr *e) override;

    int interp() override;

    bool has_variable() override;

    Expr* subst(std::string string, Expr* e) override;

    void print(std::ostream &ostream) override;

    void pretty_print_at(std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool
lastLvlMult) override;

    precedence_t get_prec() override;
};



class Var : public Expr {
public:
    std::string string_;

    explicit Var(std::string varName);

    bool equals(Expr *e) override;

    int interp() override;

    bool has_variable() override;

    Expr* subst(std::string string, Expr* e) override;

    void print(std::ostream &ostream) override;

    void pretty_print_at(std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool
lastLvlMult) override;

    precedence_t get_prec() override;
};



class Add : public Expr {
public:
    Expr *lhs_;
    Expr *rhs_;

    Add(Expr *lhs, Expr *rhs);

    bool equals(Expr *e) override;

    int interp() override;

    bool has_variable() override;

    Expr* subst(std::string string, Expr* e) override;


    void print(std::ostream &ostream) override;

    void pretty_print_at(std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool
lastLvlMult) override;

    precedence_t get_prec() override;
};



class Mult : public Expr {
public:
    Expr *lhs_;
    Expr *rhs_;

    Mult(Expr *lhs, Expr *rhs);

    bool equals(Expr *e) override;

    int interp() override;

    bool has_variable() override;

    Expr* subst(std::string string, Expr* e) override;

    void print(std::ostream &ostream) override;

    void pretty_print_at(std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool
lastLvlMult) override;
```

```

    precedence_t get_prec() override;
};

class Let : public Expr {
public:
    std::string lhs_;
    Expr *rhs_;
    Expr *body_;

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

    bool equals(Expr *e) override;

    int interp() override;

    bool has_variable() override;

    Expr* subst(std::string string, Expr* e) override;

    void print(std::ostream &ostream) override;

    void pretty_print_at(std::ostream &ostream, std::streampos &lastReturnSeen, bool lastLvlLeft, bool
lastLvlMult) override;

    precedence_t get_prec() override;
};
```

Index