My Project

Generated by Doxygen 1.9.6

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

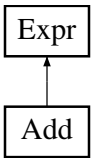
Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Add Class Reference

Inheritance diagram for Add:



Public Member Functions

- Add (Expr *lhs, Expr *rhs)
  - *constructor for Add class*
- bool equals (Expr *n) override
  - *returns if an expression is equal to another*
- int interp ()
  - *Gives the value of the number at the bottom of the expression*
- bool has_variable ()
  - *Returns if the left hand side or the right hand side expression has a variable*
- Expr * subst (std::string variable, Expr *expr)
  - *Returns an expression if the value is the variable, otherwise it returns the Var*
- void print (std::ostream &stream)
  - *Prints the left side and right side within parentheses*
- void pretty_print_at (std::ostream &stream, precedence_t prec)
  - *utilizes the precedence from the parent to print correct parentheses*

Public Member Functions inherited from Expr

- virtual bool equals (Expr ∗e)=0
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr ∗ subst (std::string variable, Expr ∗expr)=0
- virtual void print (std::ostream &stream)=0
- std::string to_string ()
  - *Returns an expression as a string*
- std::string pretty_print_to_string ()
  - *Returns an expression with correct parentheses*
- void pretty_print (std::ostream &stream)
  - *Prints to console the correct parentheses*
- virtual void pretty_print_at (std::ostream &stream, precedence_t prec)=0

Public Attributes

- Expr ∗ **lhs**
- Expr ∗ **rhs**

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Add()

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

- constructor for Add class

Parameters

lhs	- left hand side of the equation
rhs	- right hand side of the equation

4.1.2 Member Function Documentation

4.1.2.1 equals()

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

- returns if an expression is equal to another

Parameters

e	- the expression on the right

Returns

- if expression is equal to another

Implements Expr.

4.1.2.2 has_variable()

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

- Returns if the left hand side or the right hand side expression has a variable

Returns

- Returns if the left hand side or the right hand side expression has a variable

Implements Expr.

4.1.2.3 interp()

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

- Gives the value of the number at the bottom of the expression

Returns

- Returns the value of the number adding to the other bottom of the expression

Implements Expr.

4.1.2.4 pretty_print_at()

```
void Add::pretty_print_at (
            std::ostream & stream,
            precedence_t prec )  [virtual]
```

- utilizes the precedence from the parent to print correct parentheses

Parameters

stream	- the stream being utilized
prec	- the precedence is always 1 for Adds

Implements Expr.

4.1.2.5 print()

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

- Prints the left side and right side within parentheses

Parameters

stream	- the stream to be utilized

Implements Expr.

4.1.2.6 subst()

```
Expr * Add::subst (
            std::string variable,
            Expr * replacement )  [virtual]
```

- Returns an expression if the value is the variable, otherwise it returns the Var

Parameters

variable	- the variable being checked in string format
replacement	- the expression being checked

Returns

- Returns an expression if the value is the variable, otherwise it returns the Var

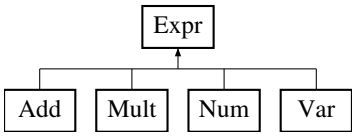
Implements Expr.

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

- /Users/jammerkoi/6015/msdscript/msdscript/Expr.h
- /Users/jammerkoi/6015/msdscript/msdscript/Expr.cpp

4.2 Expr Class Reference

Inheritance diagram for Expr:



Public Member Functions

- virtual bool equals (Expr ∗e)=0
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr ∗ subst (std::string variable, Expr ∗expr)=0
- virtual void print (std::ostream &stream)=0
- std::string to_string ()
  - *Returns an expression as a string*
- std::string pretty_print_to_string ()
  - *Returns an expression with correct parentheses*
- void pretty_print (std::ostream &stream)
  - *Prints to console the correct parentheses*
- virtual void pretty_print_at (std::ostream &stream, precedence_t prec)=0

4.2.1 Member Function Documentation

4.2.1.1 equals()

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

Implemented in Num, Var, Add, and Mult.

4.2.1.2 has_variable()

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

Implemented in Num, Var, Add, and Mult.

4.2.1.3 interp()

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

Implemented in Num, Var, Add, and Mult.

4.2.1.4 pretty_print()

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

- Prints to console the correct parentheses

Parameters

stream	- the stream input to console

4.2.1.5 pretty_print_at()

```
virtual void Expr::pretty_print_at (
             std::ostream & stream,
             precedence_t prec )  [pure virtual]
```

Implemented in Num, Var, Add, and Mult.

4.2.1.6 pretty_print_to_string()

```
std::string Expr::pretty_print_to_string ( )
```

- Returns an expression with correct parentheses

Returns

- Returns a string to be printed

4.2.1.7 print()

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

Implemented in Num, Var, Add, and Mult.

4.2.1.8 subst()

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

Implemented in Num, Var, Add, and Mult.

4.2.1.9 to_string()

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

- Returns an expression as a string

Parameters

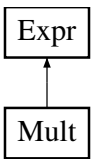
-	Takes in the expression that calls it @return - Returns an expression as a string

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

- /Users/jammerkoi/6015/msdscript/msdscript/Expr.h
- /Users/jammerkoi/6015/msdscript/msdscript/Expr.cpp

4.3 Mult Class Reference

Inheritance diagram for Mult:



Public Member Functions

- Mult (Expr *lhs, Expr *rhs)
  - *the constructor for Mult class*
- bool equals (Expr *n) override
  - *returns if an expression is equal to another*
- int interp ()
  - *Gives the value of the number at the bottom of the expression*
- bool has_variable ()
  - *Tells if the expression calling has a variable on the left or right hand side*
- Expr * subst (std::string variable, Expr *expr)
  - *Returns an expression if the value is the variable, otherwise it returns the Var*
- void print (std::ostream &stream)
  - *Prints the right hand side and left hand side with a '*' in between. The whole thing is surrounded by parentheses*
- void pretty_print_at (std::ostream &stream, precedence_t prec)
  - *utilizes the precedence from the parent to print correct parentheses*

Public Member Functions inherited from Expr

- virtual bool equals (Expr *e)=0
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr * subst (std::string variable, Expr *expr)=0
- virtual void print (std::ostream &stream)=0
- std::string to_string ()
  - *Returns an expression as a string*
- std::string pretty_print_to_string ()
  - *Returns an expression with correct parentheses*
- void pretty_print (std::ostream &stream)
  - *Prints to console the correct parentheses*
- virtual void pretty_print_at (std::ostream &stream, precedence_t prec)=0

Public Attributes

- Expr * **lhs**
- Expr * **rhs**

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Mult()

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

- the constructor for Mult class

Parameters

lhs	- the lhs expression of the equation
rhs	- the rhs expression of the equation

4.3.2 Member Function Documentation

4.3.2.1 equals()

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

- returns if an expression is equal to another

Parameters

e	- the expression on the right

Returns

- if expression is equal to another

Implements Expr.

4.3.2.2 has_variable()

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

- Tells if the expression calling has a variable on the left or right hand side

Returns

- Returns true if either expression has a Var

Implements Expr.

4.3.2.3 interp()

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

- Gives the value of the number at the bottom of the expression

Returns

- Returns the value of the number adding to the other bottom of the expression

Implements Expr.

4.3.2.4 pretty_print_at()

```
void Mult::pretty_print_at (
            std::ostream & stream,
            precedence_t prec )  [virtual]
```

- utilizes the precedence from the parent to print correct parentheses

Parameters

stream	- the stream being utilized
prec	- the precedence is always 2 for Mults

Implements Expr.

4.3.2.5 print()

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

- Prints the right hand side and left hand side with a '∗' in between. The whole thing is surrounded by parentheses

Parameters

stream	- The stream being utilized

Implements Expr.

4.3.2.6 subst()

```
Expr * Mult::subst (
            std::string variable,
            Expr * replacement )  [virtual]
```

- Returns an expression if the value is the variable, otherwise it returns the Var

Parameters

variable	- the variable being checked in string format
replacement	- the expression being checked

Returns

- Returns an expression if the value is the variable, otherwise it returns the Var

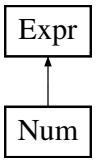
Implements Expr.

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

- /Users/jammerkoi/6015/msdscript/msdscript/Expr.h
- /Users/jammerkoi/6015/msdscript/msdscript/Expr.cpp

4.4 Num Class Reference

Inheritance diagram for Num:



Public Member Functions

- Num (int val)
  - *Constructor for a number*
- bool equals (Expr ∗n)
  - *returns if an expression is equal to another*
- int interp ()
  - *Gives the value of the number*
- bool has_variable ()
  - *Checks to see if the number has any variables which is always false*
- Expr ∗ subst (std::string variable, Expr ∗expr)
  - *Returns a number being called*
- void print (std::ostream &stream)
  - *prints the object calling to the stream*
- void pretty_print_at (std::ostream &stream, precedence_t prec)
  - *utilizes the precedence from the parent to print correct parentheses*

Public Member Functions inherited from Expr

- virtual bool equals (Expr ∗e)=0
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr ∗ subst (std::string variable, Expr ∗expr)=0
- virtual void print (std::ostream &stream)=0
- std::string to_string ()
  - *Returns an expression as a string*
- std::string pretty_print_to_string ()
  - *Returns an expression with correct parentheses*
- void pretty_print (std::ostream &stream)
  - *Prints to console the correct parentheses*
- virtual void pretty_print_at (std::ostream &stream, precedence_t prec)=0

Public Attributes

- int **val**

4.4.1 Constructor & Destructor Documentation

4.4.1.1 Num()

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

- Constructor for a number

Parameters

val	- takes in an int

4.4.2 Member Function Documentation

4.4.2.1 equals()

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

- returns if an expression is equal to another

Parameters

e	- the expression on the right

Returns

- if expression is equal to another

Implements Expr.

4.4.2.2 has_variable()

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

- Checks to see if the number has any variables which is always false

Returns

- False since a num can't have a var

Implements Expr.

4.4.2.3 interp()

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

- Gives the value of the number

Returns

- Returns the value of the number

Implements Expr.

4.4.2.4 pretty_print_at()

```
void Num::pretty_print_at (
            std::ostream & stream,
            precedence_t prec_none )  [virtual]
```

- utilizes the precedence from the parent to print correct parentheses

Parameters

stream	- the stream being utilized
prec_none	- the precedence is always zero for nums

Implements Expr.

4.4.2.5 print()

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

- prints the object calling to the stream

Parameters

stream	- the stream being utilized

Implements Expr.

4.4.2.6 subst()

```
Expr * Num::subst (
             std::string variable,
             Expr * expr )  [virtual]
```

- Returns a number being called

Parameters

variable	- the variable being checked in string format
expr	- the expression being checked

Returns

- the number being called

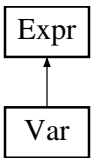
Implements Expr.

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

- /Users/jammerkoi/6015/msdscript/msdscript/Expr.h
- /Users/jammerkoi/6015/msdscript/msdscript/Expr.cpp

4.5 Var Class Reference

Inheritance diagram for Var:



Public Member Functions

- Var (std::string val)
  - *the constructor for a Var*
- bool equals (Expr ∗n) override
  - *returns if an expression is equal to another*
- int interp ()
  - *throws an exception as there is no value for Vars*
- bool has_variable ()
  - *Returns true as Var class is a var*
- Expr ∗ subst (std::string variable, Expr ∗expr)
  - *Returns an expression if the value is the variable, otherwise it returns the Var*
- void print (std::ostream &stream)
  - *prints the object calling to the stream*
- void pretty_print_at (std::ostream &stream, precedence_t prec)
  - *utilizes the precedence from the parent to print correct parentheses*

Public Member Functions inherited from Expr

- virtual bool equals (Expr ∗e)=0
- virtual int interp ()=0
- virtual bool has_variable ()=0
- virtual Expr ∗ subst (std::string variable, Expr ∗expr)=0
- virtual void print (std::ostream &stream)=0
- std::string to_string ()
  - *Returns an expression as a string*
- std::string pretty_print_to_string ()
  - *Returns an expression with correct parentheses*
- void pretty_print (std::ostream &stream)
  - *Prints to console the correct parentheses*
- virtual void pretty_print_at (std::ostream &stream, precedence_t prec)=0

Public Attributes

- std::string **val**

4.5.1 Constructor & Destructor Documentation

4.5.1.1 Var()

```
Var::Var (
            std::string val )
```

- the constructor for a Var

Parameters

val	- the string representing the var

4.5.2 Member Function Documentation

4.5.2.1 equals()

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

- returns if an expression is equal to another

Parameters

e	- the expression on the right

Returns

- if expression is equal to another

Implements Expr.

4.5.2.2 has_variable()

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

- Returns true as Var class is a var

Returns

- True

Implements Expr.

4.5.2.3 interp()

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

- throws an exception as there is no value for Vars

Returns

- exception

Implements Expr.

4.5.2.4 pretty_print_at()

```
void Var::pretty_print_at (
            std::ostream & stream,
            precedence_t prec_none )  [virtual]
```

- utilizes the precedence from the parent to print correct parentheses

Parameters

stream	- the stream being utilized
prec_none	- the precedence is always zero for nums

Implements Expr.

4.5.2.5 print()

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

- prints the object calling to the stream

Parameters

stream	- the stream being utilized

Implements Expr.

4.5.2.6 subst()

```
Expr * Var::subst (
            std::string variable,
            Expr * expr )  [virtual]
```

- Returns an expression if the value is the variable, otherwise it returns the Var

Parameters

variable	- the variable being checked in string format
expr	- the expression being checked

Returns

- Returns an expression if the value is the variable, otherwise it returns the Var

Implements Expr.

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

- /Users/jammerkoi/6015/msdscript/msdscript/Expr.h
- /Users/jammerkoi/6015/msdscript/msdscript/Expr.cpp

Chapter 5

File Documentation

5.1 /Users/jammerkoi/6015/msdscript/msdscript/cmdline.h

```
00001 //
00002 // Created by Mack on 1/13/23.
00003 //
00004
00005 #ifndef HW1_CMDLINE_H
00006 #define HW1_CMDLINE_H
00007 using namespace std;
00008
00009 int use_arguments(int argc, char** argv);
00010
00011 #endif //HW1_CMDLINE_H
```

5.2 /Users/jammerkoi/6015/msdscript/msdscript/Expr.h

```
00001 //
00002 // Created by Mack on 1/18/23.
00003 //
00004
00005 #ifndef EXPRESSIONSHW_EXPR_H
00006 #define EXPRESSIONSHW_EXPR_H
00007 #include <string>
00008 //caps
00009
00010 typedef enum {
00011
00012     prec_none,      // = 0
00013     prec_add,       // = 1
00014     prec_mult       // = 2
00015 } precedence_t;
00016
00017 class Expr {
00018
00019 public:
00020     virtual bool equals(Expr *e) = 0; // 0 means subclass must overwrite equals
00021     virtual int interp() = 0;
00022     virtual bool has_variable() = 0;
00023     virtual Expr* subst(std::string variable, Expr* expr) = 0;
00024     virtual void print(std::ostream& stream) = 0;
00025     std::string to_string();
00026     std::string pretty_print_to_string();
00027     void pretty_print(std::ostream&stream);
00028     virtual void pretty_print_at(std::ostream& stream, precedence_t prec)= 0;
00029
00030
00031
00032 };
00033
00034 class Num : public Expr {
00035 public:
00036     int val;
00037     Num(int val);
00038     bool equals(Expr *n);
00039     int interp();
00040     bool has_variable();
```

```
00041     Expr* subst(std::string variable, Expr* expr);
00042     void print(std::ostream& stream);
00043     void pretty_print_at(std::ostream& stream, precedence_t prec);
00044
00045 };
00046
00047 class Var : public Expr {
00048 public:
00049     std::string val;
00050     Var(std::string val);
00051     bool equals(Expr *n) override;
00052     int interp();
00053     bool has_variable();
00054     Expr* subst(std::string variable, Expr* expr);
00055     void print(std::ostream& stream);
00056
00057     void pretty_print_at(std::ostream& stream, precedence_t prec);
00058 };
00059
00060 class Add : public Expr {
00061 public:
00062     Expr *lhs;
00063     Expr *rhs;
00064     Add(Expr *lhs, Expr *rhs);
00065     bool equals(Expr *n) override;
00066     int interp();
00067     bool has_variable();
00068     Expr* subst(std::string variable, Expr* expr);
00069     void print(std::ostream& stream);
00070
00071     void pretty_print_at(std::ostream& stream, precedence_t prec);
00072 };
00073
00074 class Mult : public Expr {
00075 public:
00076     Expr *lhs;
00077     Expr *rhs;
00078     Mult(Expr *lhs, Expr *rhs);
00079     bool equals(Expr *n) override;
00080     int interp();
00081     bool has_variable();
00082     Expr* subst(std::string variable, Expr* expr);
00083     void print(std::ostream& stream);
00084
00085     void pretty_print_at(std::ostream& stream, precedence_t prec);
00086 };
00087
00088 #endif //EXPRESSIONSHW_EXPR_H
```

5.3 /Users/jammerkoi/6015/msdscript/msdscript/expTest.h

```
00001 //
00002 // Created by Mack on 1/19/23.
00003 //
00004
00005 #ifndef MSDSCRIPT_EXPTEST_H
00006 #define MSDSCRIPT_EXPTEST_H
00007 #include "catch.h"
00008 #include "Expr.h"
00009
00018 TEST_CASE("interp")
00019 {
00020     CHECK( (new Mult(new Num(3), new Num(2)))
00021                     ->interp()==6 );
00022     CHECK( (new Add(new Add(new Num(10), new Num(15)),new Add(new Num(20),new Num(20))))
00023                     ->interp()==65);
00024 }
00025
00026 TEST_CASE( "equals" ) {
00027     CHECK((new Num(1))->equals(new Num(1)) == true );
00028     CHECK((new Var("x"))->equals(new Var("y")) == false );
00029     CHECK((new Add(new Num(2), new Num(3)))->equals(new Add(new  Num(2), new Num(3))) == true );
00030 }
00031
00032 //    TEST_CASE( "not equals" ) {
00033 //    CHECK((new Mult(new Num(1), new Num(2)))->equals(new Add(new Num(1), new Num(2))) == false);
00034 //    CHECK((new Add(new Num(2), new Num(3)))->equals(new Add(new Num(3), new Num(2))) == false);
00035 //
00036 //}
00037
00038 TEST_CASE( "different types same Num vals" ) {
00039     CHECK((new Add(new Num(2), new Num(24)))->equals(new Mult(new Num(2), new Num(24))) == false);
00040     CHECK((new Add(new Num(-143), new Num(25)))->equals(new Mult(new Num(-143), new Num(25))) ==
      false);
```

```
00041     CHECK_FALSE((new Num(1))->equals(new Num(5)));
00042 //    CHECK_FALSE((new Num(1))->equals(new Num(5)));
00043     CHECK((new Mult(new Mult(new Num(4), new Num(6)), new Num(5)), new Num(55))->equals
00044           ((new Mult(new Mult(new Num(4), new Num(6)), new Num(5)), new Num(55))) == true);
00045 }
00046
00047 TEST_CASE("checking sub for add") {
00048     CHECK( (new Add(new Var("x"), new Num(24)))
00049                     ->subst("x", new Var("y"))
00050                     ->equals(new Add(new Var("y"), new Num(24))) );
00051
00052     CHECK( (new Add(new Var("x"), new Num(7)))
00053                     ->subst("x", new Var("y"))
00054                     ->equals(new Add(new Var("y"), new Num(7))) );
00055
00056     CHECK( (new Var("x"))
00057                     ->subst("x", new Add(new Var("y"),new Num(7)))
00058                     ->equals(new Add(new Var("y"),new Num(7))));
00059
00060
00061 }
00062
00063 TEST_CASE("Checking mult expression substitution") {
00064     CHECK( (new Mult(new Var("x"), new Num(24)))
00065                     ->subst("x", new Var("y"))
00066                     ->equals(new Mult(new Var("y"), new Num(24))) );
00067
00068     CHECK_FALSE( (new Mult(new Var("x"), new Num(24)))
00069                     ->subst("z", new Var("y"))
00070                     ->equals(new Mult(new Var("y"), new Num(24))) );
00071 }
00072
00073 TEST_CASE("Checking substituting something that isn't in there") {
00074     CHECK( (new Add(new Var("x"), new Num(24)))
00075                     ->subst("z", new Var("y"))
00076                     ->equals(new Add(new Var("x"), new Num(24))) );
00077
00078     CHECK( (new Mult(new Var("x"), new Num(24)))
00079                     ->subst("z", new Var("y"))
00080                     ->equals(new Mult(new Var("x"), new Num(24))) );
00081 }
00082
00083 TEST_CASE("CHECKING PRINT") {
00084     CHECK( (new Num(10))->to_string() == "10" );
00085 }
00086
00087 TEST_CASE("CHECKING PRETTY PRINT") {
00088 //    new Mult (new Mult (new Num(2), new Num(3)), new Num(4))->equals()
00089 //    CHECK(((new Mult( new Num(1), new Add(new Num(2), new Num(3)))->pretty_print_to_string()) == "1
      * (2 + 3)");
00090     //->to_string()->equals("1 * (2+3")));
00091     CHECK( (new Mult (new Mult (new Num(2), new Num(3)), new Num(4)))->pretty_print_to_string() == "(2
      * 3) * 4");
00092     CHECK((new Num(1))->pretty_print_to_string() == "1");
00093     CHECK((new Var("a"))->pretty_print_to_string() == "a");
00094     CHECK((new Add( new Num(1), new Num(3)))->pretty_print_to_string() == "1 + 3");
00095
00096
00097 }
00098
00099
00100
00101 #endif MSDSCRIPT_EXPTEST_H
```

Index