Brandon's MSD Script

Generated by Doxygen 1.13.2

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

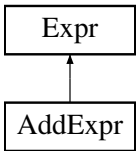
Class Documentation

4.1 AddExpr Class Reference

Represents an addition expression.

`#include <expr.h>`

Inheritance diagram for AddExpr:



Public Member Functions

- AddExpr (Expr ∗lhs, Expr ∗rhs)

  Constructs an addition expression.
- bool equals (const Expr ∗e) override

  Checks if this addition expression is equal to another expression.
- int interp () override

  Evaluates the addition expression to its result.
- bool has_variable () override

  Checks if the addition expression contains a variable.
- Expr ∗ subst (const std::string &var, Expr ∗replacement) override

  Substitutes a variable with another expression in the addition expression.
- void printExp (std::ostream &ot) override

  Prints the addition expression to an output stream.
- void pretty_print (std::ostream &ot, precedence_t prec) override

  Pretty-prints the addition expression to an output stream with proper precedence handling.

Public Member Functions inherited from Expr

- std::string to_string ()

  Converts the expression to a string.
- std::string to_pretty_string ()

  Converts the expression to a pretty-printed string.

4.1.1 Detailed Description

Represents an addition expression.

This class represents an expression that adds two sub-expressions.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AddExpr()

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

Constructs an addition expression.

Parameters

lhs	The left-hand side expression.
rhs	The right-hand side expression.

4.1.3 Member Function Documentation

4.1.3.1 equals()

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

Checks if this addition expression is equal to another expression.

Parameters

e	The expression to compare with.

Returns

true if the expressions are equal, false otherwise.

Implements Expr.

4.1.3.2 has_variable()

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

Checks if the addition expression contains a variable.

Returns

true if either the left-hand side or right-hand side contains a variable, false otherwise.

Implements Expr.

4.1.3.3 interp()

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

Evaluates the addition expression to its result.

Returns

The sum of the left-hand side and right-hand side expressions.

Implements Expr.

4.1.3.4 pretty_print()

```
void AddExpr::pretty_print (
            std::ostream & ot,
            precedence_t prec)  [override], [virtual]
```

Pretty-prints the addition expression to an output stream with proper precedence handling.

Parameters

ot	The output stream to print to.
prec	The precedence level of the parent expression.

Implements Expr.

4.1.3.5 printExp()

```
void AddExpr::printExp (
            std::ostream & ot)  [override], [virtual]
```

Prints the addition expression to an output stream.

Parameters

ot	The output stream to print to.

Implements Expr.

4.1.3.6 subst()

```
Expr * AddExpr::subst (
            const std::string & var,
            Expr * replacement)  [override], [virtual]
```

Substitutes a variable with another expression in the addition expression.

Parameters

var	The variable to substitute.
replacement	The expression to replace the variable with.

Returns

A new addition expression with the substitution applied.

Implements Expr.

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

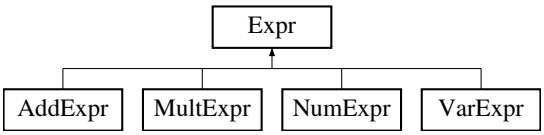
- /Users/brandonmountan/MSD/CS6015_SoftwareEngineering/Assignments/Project_1/expr.h
- /Users/brandonmountan/MSD/CS6015_SoftwareEngineering/Assignments/Project_1/expr.cpp

4.2 Expr Class Reference

Abstract base class for expressions.

`#include <expr.h>`

Inheritance diagram for Expr:



Public Member Functions

- virtual bool equals (const Expr *e)=0

  Checks if this expression is equal to another expression.
- virtual int interp ()=0

  Evaluates the expression to an integer value.
- virtual bool has_variable ()=0

  Checks if the expression contains a variable.
- virtual Expr * subst (const std::string &var, Expr *replacement)=0

  Substitutes a variable with another expression.
- virtual void printExp (std::ostream &ot)=0

  Prints the expression to an output stream.
- std::string to_string ()

  Converts the expression to a string.
- virtual void pretty_print (std::ostream &ot, precedence_t prec)=0

  Pretty-prints the expression to an output stream with proper precedence handling.
- std::string to_pretty_string ()

  Converts the expression to a pretty-printed string.

4.2.1 Detailed Description

Abstract base class for expressions.

This class defines the interface for all expression types. It includes pure virtual methods that must be implemented by derived classes.

4.2.2 Member Function Documentation

4.2.2.1 equals()

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

Checks if this expression is equal to another expression.

Parameters

e	The expression to compare with.

Returns

true if the expressions are equal, false otherwise.

Implemented in AddExpr, MultExpr, NumExpr, and VarExpr.

4.2.2.2 has_variable()

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

Checks if the expression contains a variable.

Returns

true if the expression contains a variable, false otherwise.

Implemented in AddExpr, MultExpr, NumExpr, and VarExpr.

4.2.2.3 interp()

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

Evaluates the expression to an integer value.

Returns

The result of evaluating the expression.

Implemented in AddExpr, MultExpr, NumExpr, and VarExpr.

4.2.2.4 pretty_print()

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

Pretty-prints the expression to an output stream with proper precedence handling.

Parameters

ot	The output stream to print to.
prec	The precedence level of the parent expression.

Implemented in AddExpr, MultExpr, NumExpr, and VarExpr.

4.2.2.5 printExp()

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

Prints the expression to an output stream.

Parameters

ot	The output stream to print to.

Implemented in AddExpr, MultExpr, NumExpr, and VarExpr.

4.2.2.6 subst()

```
virtual Expr * Expr::subst (
            const std::string & var,
            Expr * replacement)  [pure virtual]
```

Substitutes a variable with another expression.

Parameters

var	The variable to substitute.
replacement	The expression to replace the variable with.

Returns

A new expression with the substitution applied.

Implemented in AddExpr, MultExpr, NumExpr, and VarExpr.

4.2.2.7 to_pretty_string()

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

Converts the expression to a pretty-printed string.

Returns

A pretty-printed string representation of the expression.

4.2.2.8 to_string()

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

Converts the expression to a string.

Returns

A string representation of the expression.

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

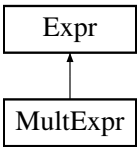
- /Users/brandonmountan/MSD/CS6015_SoftwareEngineering/Assignments/Project_1/expr.h
- /Users/brandonmountan/MSD/CS6015_SoftwareEngineering/Assignments/Project_1/expr.cpp

4.3 MultExpr Class Reference

Represents a multiplication expression.

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

Inheritance diagram for MultExpr:



Public Member Functions

- MultExpr (Expr *lhs, Expr *rhs)
  Constructs a multiplication expression.
- bool equals (const Expr *e) override
  Checks if this multiplication expression is equal to another expression.
- int interp () override
  Evaluates the multiplication expression to its result.
- bool has_variable () override
  Checks if the multiplication expression contains a variable.
- Expr * subst (const std::string &var, Expr *replacement) override
  Substitutes a variable with another expression in the multiplication expression.
- void printExp (std::ostream &ot) override
  Prints the multiplication expression to an output stream.
- void pretty_print (std::ostream &ot, precedence_t prec) override
  Pretty-prints the multiplication expression to an output stream with proper precedence handling.

Public Member Functions inherited from Expr

- std::string to_string ()

  Converts the expression to a string.
- std::string to_pretty_string ()

  Converts the expression to a pretty-printed string.

4.3.1 Detailed Description

Represents a multiplication expression.

This class represents an expression that multiplies two sub-expressions.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 MultExpr()

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

Constructs a multiplication expression.

Parameters

lhs	The left-hand side expression.
rhs	The right-hand side expression.

4.3.3 Member Function Documentation

4.3.3.1 equals()

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

Checks if this multiplication expression is equal to another expression.

Parameters

e	The expression to compare with.

Returns

true if the expressions are equal, false otherwise.

Implements Expr.

4.3.3.2 has_variable()

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

Checks if the multiplication expression contains a variable.

Returns

true if either the left-hand side or right-hand side contains a variable, false otherwise.

Implements Expr.

4.3.3.3 interp()

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

Evaluates the multiplication expression to its result.

Returns

The product of the left-hand side and right-hand side expressions.

Implements Expr.

4.3.3.4 pretty_print()

```
void MultExpr::pretty_print (
            std::ostream & ot,
            precedence_t prec)  [override], [virtual]
```

Pretty-prints the multiplication expression to an output stream with proper precedence handling.

Parameters

ot	The output stream to print to.
prec	The precedence level of the parent expression.

Implements Expr.

4.3.3.5 printExp()

```
void MultExpr::printExp (
            std::ostream & ot)  [override], [virtual]
```

Prints the multiplication expression to an output stream.

Parameters

ot	The output stream to print to.

Implements Expr.

4.3.3.6 subst()

```
Expr * MultExpr::subst (
            const std::string & var,
            Expr * replacement)  [override], [virtual]
```

Substitutes a variable with another expression in the multiplication expression.

Parameters

var	The variable to substitute.
replacement	The expression to replace the variable with.

Returns

A new multiplication expression with the substitution applied.

Implements Expr.

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

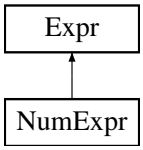
- /Users/brandonmountan/MSD/CS6015_SoftwareEngineering/Assignments/Project_1/expr.h
- /Users/brandonmountan/MSD/CS6015_SoftwareEngineering/Assignments/Project_1/expr.cpp

4.4 NumExpr Class Reference

Represents a numeric expression.

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

Inheritance diagram for NumExpr:



Public Member Functions

- NumExpr (int value)
  Constructs a numeric expression.
- bool equals (const Expr *e) override
  Checks if this numeric expression is equal to another expression.
- int interp () override
  Evaluates the numeric expression to its value.
- bool has_variable () override
  Checks if the numeric expression contains a variable.
- Expr * subst (const std::string &var, Expr *replacement) override
  Substitutes a variable with another expression (no effect for numeric expressions).
- void printExp (std::ostream &ot) override
  Prints the numeric expression to an output stream.
- void pretty_print (std::ostream &ot, precedence_t prec) override
  Pretty-prints the numeric expression to an output stream.

Public Member Functions inherited from Expr

- std::string to_string ()
  Converts the expression to a string.
- std::string to_pretty_string ()
  Converts the expression to a pretty-printed string.

4.4.1 Detailed Description

Represents a numeric expression.

This class represents an expression that consists of a single numeric value.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 NumExpr()

```
NumExpr::NumExpr (
            int value)
```

Constructs a numeric expression.

Parameters

value	The numeric value.

4.4.3 Member Function Documentation

4.4.3.1 equals()

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

Checks if this numeric expression is equal to another expression.

Parameters

e	The expression to compare with.

Returns

true if the expressions are equal, false otherwise.

Implements Expr.

4.4.3.2 has_variable()

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

Checks if the numeric expression contains a variable.

Returns

false (numeric expressions do not contain variables).

Implements Expr.

4.4.3.3 interp()

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

Evaluates the numeric expression to its value.

Returns

The numeric value.

Implements Expr.

4.4.3.4 pretty_print()

```
void NumExpr::pretty_print (
            std::ostream & ot,
            precedence_t prec)  [override], [virtual]
```

Pretty-prints the numeric expression to an output stream.

Parameters

ot	The output stream to print to.
prec	The precedence level of the parent expression.

Implements Expr.

4.4.3.5 printExp()

```
void NumExpr::printExp (
            std::ostream & ot)  [override], [virtual]
```

Prints the numeric expression to an output stream.

Parameters

ot	The output stream to print to.

Implements Expr.

4.4.3.6 subst()

```
Expr * NumExpr::subst (
            const std::string & var,
            Expr * replacement)  [override], [virtual]
```

Substitutes a variable with another expression (no effect for numeric expressions).

Parameters

var	The variable to substitute.
replacement	The expression to replace the variable with.

Returns

This numeric expression (no substitution occurs).

Implements Expr.

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

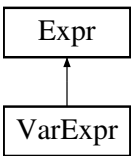
- /Users/brandonmountan/MSD/CS6015_SoftwareEngineering/Assignments/Project_1/expr.h
- /Users/brandonmountan/MSD/CS6015_SoftwareEngineering/Assignments/Project_1/expr.cpp

4.5 VarExpr Class Reference

Represents a variable expression.

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

Inheritance diagram for VarExpr:



Public Member Functions

- VarExpr (const std::string &name)
  Constructs a variable expression.
- bool equals (const Expr *e) override
  Checks if this variable expression is equal to another expression.
- int interp () override
  Evaluates the variable expression (throws an exception since variables have no value).
- bool has_variable () override
  Checks if the variable expression contains a variable.
- Expr * subst (const std::string &var, Expr *replacement) override
  Substitutes a variable with another expression in the variable expression.
- void printExp (std::ostream &ot) override
  Prints the variable expression to an output stream.
- void pretty_print (std::ostream &ot, precedence_t prec) override
  Pretty-prints the variable expression to an output stream.

Public Member Functions inherited from Expr

- std::string to_string ()
  Converts the expression to a string.
- std::string to_pretty_string ()
  Converts the expression to a pretty-printed string.

4.5.1 Detailed Description

Represents a variable expression.

This class represents an expression that consists of a variable.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 VarExpr()

```
VarExpr::VarExpr (
            const std::string & name)
```

Constructs a variable expression.

Parameters

name	The name of the variable.

4.5.3 Member Function Documentation

4.5.3.1 equals()

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

Checks if this variable expression is equal to another expression.

Parameters

e	The expression to compare with.

Returns

true if the expressions are equal, false otherwise.

Implements Expr.

4.5.3.2 has_variable()

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

Checks if the variable expression contains a variable.

Returns

true (variable expressions always contain a variable).

Implements Expr.

4.5.3.3 interp()

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

Evaluates the variable expression (throws an exception since variables have no value).

Exceptions

std::runtime_error	Always throws an exception.

Implements Expr.

4.5.3.4 pretty_print()

```
void VarExpr::pretty_print (
            std::ostream & ot,
            precedence_t prec)  [override], [virtual]
```

Pretty-prints the variable expression to an output stream.

Parameters

ot	The output stream to print to.
prec	The precedence level of the parent expression.

Implements Expr.

4.5.3.5 printExp()

```
void VarExpr::printExp (
            std::ostream & ot)  [override], [virtual]
```

Prints the variable expression to an output stream.

Parameters

ot	The output stream to print to.

Implements Expr.

4.5.3.6 subst()

```
Expr * VarExpr::subst (
            const std::string & var,
            Expr * replacement)  [override], [virtual]
```

Substitutes a variable with another expression in the variable expression.

Parameters

var	The variable to substitute.
replacement	The expression to replace the variable with.

Returns

The replacement expression if the variable matches, otherwise this variable expression.

Implements Expr.

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

- /Users/brandonmountan/MSD/CS6015_SoftwareEngineering/Assignments/Project_1/expr.h
- /Users/brandonmountan/MSD/CS6015_SoftwareEngineering/Assignments/Project_1/expr.cpp

Chapter 5

File Documentation

5.1 /Users/brandonmountan/MSD/CS6015_SoftwareEngineering/↩ Assignments/Project_1/cmdline.h

```
00001 //
00002 // Created by Brandon Mountan on 1/13/25.
00003 //
00004
00005 #ifndef CMDLINE_H
00006 #define CMDLINE_H
00007
00008 void use_arguments(int argc, char *argv[]);
00009
00010 #endif //CMDLINE_H
```

5.2 /Users/brandonmountan/MSD/CS6015_SoftwareEngineering/↩ Assignments/Project_1/expr.h

```
00001 //
00002 // Created by Brandon Mountan on 1/20/25.
00003 //
00004
00005 #ifndef EXPR_H
00006 #define EXPR_H
00007
00008 #include <string>
00009 #include <stdexcept> // For std::runtime_error
00010 #include <sstream>   // For std::stringstream
00011
00019 typedef enum {
00020     prec_none,
00021     prec_add,
00022     prec_mult
00023 } precedence_t;
00024
00032 class Expr {
00033 public:
00040     virtual bool equals(const Expr* e) = 0;
00041
00047     virtual int interp() = 0;
00048
00054     virtual bool has_variable() = 0;
00055
00063     virtual Expr* subst(const std::string& var, Expr* replacement) = 0;
00064
00070     virtual void printExp(std::ostream& ot) = 0;
00071
00077     std::string to_string();
00078
00085     virtual void pretty_print(std::ostream& ot, precedence_t prec) = 0;
00086
00092     std::string to_pretty_string();
00093 };
```

```
00094
00101 class NumExpr : public Expr {
00102     int value;
00103
00104 public:
00110     NumExpr(int value);
00111
00112     bool equals(const Expr* e) override;
00113     int interp() override;
00114     bool has_variable() override;
00115     Expr* subst(const std::string& var, Expr* replacement) override;
00116
00117     void printExp(std::ostream& ot) override;
00118     void pretty_print(std::ostream& ot, precedence_t prec) override;
00119 };
00120
00127 class AddExpr : public Expr {
00128     Expr* lhs;
00129     Expr* rhs;
00130
00131 public:
00138     AddExpr(Expr* lhs, Expr* rhs);
00139
00140     bool equals(const Expr* e) override;
00141     int interp() override;
00142     bool has_variable() override;
00143     Expr* subst(const std::string& var, Expr* replacement) override;
00144
00145     void printExp(std::ostream& ot) override;
00146     void pretty_print(std::ostream& ot, precedence_t prec) override;
00147 };
00148
00155 class MultExpr : public Expr {
00156     Expr* lhs;
00157     Expr* rhs;
00158
00159 public:
00166     MultExpr(Expr* lhs, Expr* rhs);
00167
00168     bool equals(const Expr* e) override;
00169     int interp() override;
00170     bool has_variable() override;
00171     Expr* subst(const std::string& var, Expr* replacement) override;
00172
00173     void printExp(std::ostream& ot) override;
00174     void pretty_print(std::ostream& ot, precedence_t prec) override;
00175 };
00176
00183 class VarExpr : public Expr {
00184     std::string name;
00185
00186 public:
00192     VarExpr(const std::string& name);
00193
00194     bool equals(const Expr* e) override;
00195     int interp() override;
00196     bool has_variable() override;
00197     Expr* subst(const std::string& var, Expr* replacement) override;
00198
00199     void printExp(std::ostream& ot) override;
00200     void pretty_print(std::ostream& ot, precedence_t prec) override;
00201 };
00202
00203 #endif // EXPR_H
```