

MSDScript

Generated by Doxygen 1.9.6

1 MSDScript	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 AddExpr Class Reference	9
5.1.1 Constructor & Destructor Documentation	10
5.1.1.1 AddExpr()	10
5.1.2 Member Function Documentation	10
5.1.2.1 equals()	10
5.1.2.2 has_variable()	11
5.1.2.3 interp()	11
5.1.2.4 pretty_print()	11
5.1.2.5 print()	12
5.1.2.6 subst()	12
5.1.3 Member Data Documentation	12
5.1.3.1 lhs	12
5.1.3.2 rhs	13
5.2 Expr Class Reference	13
5.2.1 Member Enumeration Documentation	13
5.2.1.1 precedence_t	13
5.2.2 Member Function Documentation	14
5.2.2.1 equals()	14
5.2.2.2 has_variable()	14
5.2.2.3 interp()	15
5.2.2.4 pretty_print()	15
5.2.2.5 pretty_print_at()	15
5.2.2.6 pretty_to_string()	16
5.2.2.7 print()	16
5.2.2.8 subst()	16
5.2.2.9 to_string()	17
5.3 MultExpr Class Reference	17
5.3.1 Constructor & Destructor Documentation	18
5.3.1.1 MultExpr()	18
5.3.2 Member Function Documentation	18
5.3.2.1 equals()	18
5.3.2.2 has_variable()	19

5.3.2.3 interp()	19
5.3.2.4 pretty_print()	19
5.3.2.5 print()	20
5.3.2.6 subst()	20
5.3.3 Member Data Documentation	20
5.3.3.1 lhs	20
5.3.3.2 rhs	21
5.4 NumExpr Class Reference	21
5.4.1 Constructor & Destructor Documentation	22
5.4.1.1 NumExpr()	22
5.4.2 Member Function Documentation	22
5.4.2.1 equals()	22
5.4.2.2 has_variable()	23
5.4.2.3 interp()	23
5.4.2.4 pretty_print()	23
5.4.2.5 print()	23
5.4.2.6 subst()	24
5.4.3 Member Data Documentation	24
5.4.3.1 val	24
5.5 VarExpr Class Reference	24
5.5.1 Constructor & Destructor Documentation	25
5.5.1.1 VarExpr()	25
5.5.2 Member Function Documentation	26
5.5.2.1 equals()	26
5.5.2.2 has_variable()	26
5.5.2.3 interp()	26
5.5.2.4 pretty_print()	27
5.5.2.5 print()	27
5.5.2.6 subst()	27
5.5.3 Member Data Documentation	28
5.5.3.1 val	28
6 File Documentation	29
6.1 /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/cmdline.cpp File Reference	29
6.1.1 Detailed Description	29
6.1.2 Function Documentation	29
6.1.2.1 use_arguments()	29
6.2 /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/cmdline.h File Reference	30
6.2.1 Detailed Description	30
6.2.2 Function Documentation	30
6.2.2.1 use_arguments()	30
6.3 /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/cmdline.h	30

6.4 /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/expr.cpp File Reference	31
6.4.1 Detailed Description	31
6.5 /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/expr.h File Reference	31
6.5.1 Detailed Description	31
6.6 /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/expr.h	32
6.7 /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/main.cpp File Reference	33
6.7.1 Detailed Description	33
6.7.2 Macro Definition Documentation	34
6.7.2.1 CATCH_CONFIG_RUNNER	34
6.7.3 Function Documentation	34
6.7.3.1 main()	34
6.8 /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/testExpr.cpp File Reference	34
6.8.1 Function Documentation	34
6.8.1.1 TEST_CASE() [1/6]	34
6.8.1.2 TEST_CASE() [2/6]	35
6.8.1.3 TEST_CASE() [3/6]	35
6.8.1.4 TEST_CASE() [4/6]	35
6.8.1.5 TEST_CASE() [5/6]	35
6.8.1.6 TEST_CASE() [6/6]	35
Index	37

Chapter 1

MSDScript

Author

Zidong Fan

Date

01/20/2023

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Expr	13
AddExpr	9
MultExpr	17
NumExpr	21
VarExpr	24

Chapter 3

Class Index

3.1 Class List

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

AddExpr	9
Expr	13
MultExpr	17
NumExpr	21
VarExpr	24

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/cmdline.cpp	
Use_arguments function	29
/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/cmdline.h	
Use_arguments function	30
/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/expr.cpp	
All constructions and methods in expr class	31
/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/expr.h	
Expr class contains NumExpr, AddExpr, MultExpr, VarExpr subclasses	31
/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/main.cpp	
Int main() to run the catch tests	33
/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/testExpr.cpp	34

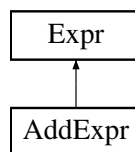
Chapter 5

Class Documentation

5.1 AddExpr Class Reference

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

Inheritance diagram for AddExpr:



Public Member Functions

- [AddExpr](#) ([Expr](#) *lhs, [Expr](#) *rhs)
- bool [equals](#) ([Expr](#) *e)
- int [interp](#) ()
- bool [has_variable](#) ()
- [Expr](#) * [subst](#) (std::string s, [Expr](#) *e)
- void [print](#) (std::ostream &ot)
- void [pretty_print](#) (std::ostream &ot)

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 s, [Expr](#) *e)=0
- virtual void [print](#) (std::ostream &ot)=0
- std::string [to_string](#) ()
- virtual void [pretty_print](#) (std::ostream &ot)=0
- std::string [pretty_to_string](#) ()
- std::string [pretty_print_at](#) (precedence_t p)

Public Attributes

- [Expr](#) * lhs
- [Expr](#) * rhs

Additional Inherited Members

Public Types inherited from [Expr](#)

- enum [precedence_t](#) { [prec_none](#) , [prec_add](#) , [prec_mult](#) }

5.1.1 Constructor & Destructor Documentation

5.1.1.1 AddExpr()

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

constructor

Parameters

<i>lhs</i>	left hand side expression
<i>rhs</i>	right hand side expression

5.1.2 Member Function Documentation

5.1.2.1 equals()

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

if input [Expr](#) is a Add and the lhs and rhs equal

Parameters

<i>e</i>	input expression for comparing
----------	--------------------------------

Returns

true if `e` is [AddExpr](#) and `e->lhs` equals to `this->lhs` and `e->rhs` equals `this->rhs`, otherwise false

Implements [Expr](#).

5.1.2.2 has_variable()

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

the add expression may contains a variable

Returns

true if [AddExpr](#) contains a variable, otherwise false

Implements [Expr](#).

5.1.2.3 interp()

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

returns an int for the value of an addition expression is the sum of the subexpression values

Returns

int

Implements [Expr](#).

5.1.2.4 pretty_print()

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

includes a space around `+` or `*`, and it avoids unnecessary parentheses by relying the usual precedence rules for multiplication and addition by assuming that operators associate to the right

Parameters

<i>ot</i>	ostream
-----------	---------

Implements [Expr](#).

5.1.2.5 print()

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

addition expressions with an infix +

Parameters

<i>ot</i>	ostream
-----------	---------

Implements [Expr](#).

5.1.2.6 subst()

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

everywhere that the expression (whose subst method is called) contains a variable matching the string, the result [Expr](#)* should have the given replacement, instead.

Parameters

<i>s</i>	a std::string is substituted
<i>e</i>	a new expression

Returns

[Expr](#)

Implements [Expr](#).

5.1.3 Member Data Documentation

5.1.3.1 lhs

```
Expr* AddExpr::lhs
```

5.1.3.2 rhs

```
Expr* AddExpr::rhs
```

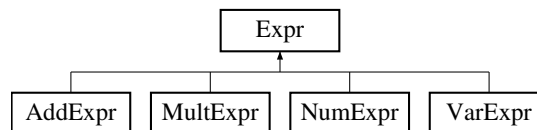
The documentation for this class was generated from the following files:

- /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/[expr.h](#)
- /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/[expr.cpp](#)

5.2 Expr Class Reference

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

Inheritance diagram for Expr:



Public Types

- enum [precedence_t](#) { [prec_none](#) , [prec_add](#) , [prec_mult](#) }

Public Member Functions

- virtual bool [equals](#) ([Expr](#) *e)=0
- virtual int [interp](#) ()=0
- virtual bool [has_variable](#) ()=0
- virtual [Expr](#) * [subst](#) (std::string s, [Expr](#) *e)=0
- virtual void [print](#) (std::ostream &ot)=0
- std::string [to_string](#) ()
- virtual void [pretty_print](#) (std::ostream &ot)=0
- std::string [pretty_to_string](#) ()
- std::string [pretty_print_at](#) ([precedence_t](#) p)

5.2.1 Member Enumeration Documentation

5.2.1.1 precedence_t

```
enum Expr::precedence_t
```

Enumerator

prec_none	
prec_add	
prec_mult	

5.2.2 Member Function Documentation

5.2.2.1 equals()

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

compare two expressions for equality

Parameters

e	input expression for comparing
---	--------------------------------

Returns

true if two expressions equal, false otherwise

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

5.2.2.2 has_variable()

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

is the expression a variable or contains a variable

Returns

true if the expression is a variable or contains a variable, false otherwise

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

5.2.2.3 interp()

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

returns an integer for the value of an expression

Returns

int

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

5.2.2.4 pretty_print()

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

virtual pretty_print: likes print, print the expression. unlike print, pretty_print includes a space around + or *, and it avoids unnecessary parentheses by relying the usual precedence rules for multiplication and addition. It can also avoid parentheses by assuming that operators associate to the right

Parameters

<i>ot</i>	ostream
-----------	---------

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

5.2.2.5 pretty_print_at()

```
std::string Expr::pretty_print_at (
    precedence_t p )
```

helper function for pretty print

Parameters

<i>p</i>	precedence type
----------	-----------------

Returns

string, how the expression print

5.2.2.6 pretty_to_string()

```
std::string Expr::pretty_to_string ( )
```

for pretty_print, should be implemented in all sub classes and should call the print method

Returns

st.str() string gonna be printed

5.2.2.7 print()

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

print the expression

Parameters

<i>ot</i>	ostream
-----------	---------

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

5.2.2.8 subst()

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

everywhere that the expression (whose subst method is called) contains a variable matching the string, the result Expr* should have the given replacement, instead.

Parameters

<i>s</i>	a std::string is substituted
<i>e</i>	a new expression

Returns

[Expr](#)

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

5.2.2.9 to_string()

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

should be implemented in all sub classes and should call the print method

Returns

st.str(), the string gonna be printed

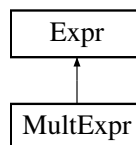
The documentation for this class was generated from the following files:

- /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/[expr.h](#)
- /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/[expr.cpp](#)

5.3 MultExpr Class Reference

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

Inheritance diagram for MultExpr:



Public Member Functions

- [MultExpr](#) ([Expr](#) *lhs, [Expr](#) *rhs)
- bool [equals](#) ([Expr](#) *e)
- int [interp](#) ()
- bool [has_variable](#) ()
- [Expr](#) * [subst](#) (std::string s, [Expr](#) *e)
- void [print](#) (std::ostream &ot)
- void [pretty_print](#) (std::ostream &ot)

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 s, [Expr](#) *e)=0
- virtual void [print](#) (std::ostream &ot)=0
- std::string [to_string](#) ()
- virtual void [pretty_print](#) (std::ostream &ot)=0
- std::string [pretty_to_string](#) ()
- std::string [pretty_print_at](#) ([precedence_t](#) p)

Public Attributes

- [Expr](#) * lhs
- [Expr](#) * rhs

Additional Inherited Members

Public Types inherited from [Expr](#)

- enum [precedence_t](#) { [prec_none](#) , [prec_add](#) , [prec_mult](#) }

5.3.1 Constructor & Destructor Documentation

5.3.1.1 MultExpr()

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

constructor

Parameters

<i>lhs</i>	left hand side expression
<i>rhs</i>	right hand side expression

5.3.2 Member Function Documentation

5.3.2.1 equals()

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

if input [Expr](#) is a Mult and the lhs and rhs equal

Parameters

<i>e</i>	input expression for comparing
----------	--------------------------------

Returns

true if `e` is [MultExpr](#) and `e->lhs` equals to `this->lhs` and `e->rhs` equals `this->rhs`, otherwise false

Implements [Expr](#).

5.3.2.2 has_variable()

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

the mult expression may contains a variable

Returns

true if [MultExpr](#) contains a variable, otherwise false

Implements [Expr](#).

5.3.2.3 interp()

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

returns an int for the value of a multiplication expression is the product of the subexpression values

Returns

int

Implements [Expr](#).

5.3.2.4 pretty_print()

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

includes a space around `+` or `*`, and it avoids unnecessary parentheses, by relying the usual precedence rules for multiplication and addition, by assuming that operators associate to the right

Parameters

<i>ot</i>	ostream
-----------	---------

Implements [Expr](#).

5.3.2.5 print()

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

multiplication expressions with an infix *

Parameters

<i>ot</i>	ostream
-----------	---------

Implements [Expr](#).

5.3.2.6 subst()

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

[MultExpr](#) subst: everywhere that the expression (whose subst method is called) contains a variable matching the string, the result [Expr](#)* should have the given replacement, instead.

Parameters

<i>s</i>	a std::string is substituted
<i>e</i>	a new expression

Returns

[Expr](#)

Implements [Expr](#).

5.3.3 Member Data Documentation

5.3.3.1 lhs

```
Expr* MultExpr::lhs
```

5.3.3.2 rhs

```
Expr* MultExpr::rhs
```

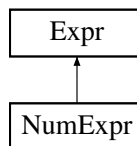
The documentation for this class was generated from the following files:

- /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/[expr.h](#)
- /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/[expr.cpp](#)

5.4 NumExpr Class Reference

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

Inheritance diagram for NumExpr:



Public Member Functions

- [NumExpr](#) (int [val](#))
- bool [equals](#) ([Expr](#) *e)
- int [interp](#) ()
- bool [has_variable](#) ()
- [Expr](#) * [subst](#) (std::string s, [Expr](#) *e)
- void [print](#) (std::ostream &ot)
- void [pretty_print](#) (std::ostream &ot)

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 s, [Expr](#) *e)=0
- virtual void [print](#) (std::ostream &ot)=0
- std::string [to_string](#) ()
- virtual void [pretty_print](#) (std::ostream &ot)=0
- std::string [pretty_to_string](#) ()
- std::string [pretty_print_at](#) ([precedence_t](#) p)

Public Attributes

- int [val](#)

Additional Inherited Members

Public Types inherited from [Expr](#)

- enum [precedence_t](#) { [prec_none](#) , [prec_add](#) , [prec_mult](#) }

5.4.1 Constructor & Destructor Documentation

5.4.1.1 NumExpr()

```
NumExpr::NumExpr (
    int val )
```

constructor

Parameters

<i>val</i>	int val
------------	---------

5.4.2 Member Function Documentation

5.4.2.1 equals()

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

if input [Expr](#) is a Num and the val equal

Parameters

<i>e</i>	input expression for comparing
----------	--------------------------------

Returns

true if e is [NumExpr](#) and e->val equals to this->val, false otherwise

Implements [Expr](#).

5.4.2.2 has_variable()

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

the number expression is not contains a variable

Returns

false

Implements [Expr](#).

5.4.2.3 interp()

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

returns an int for the value of a number is the number

Returns

int

Implements [Expr](#).

5.4.2.4 pretty_print()

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

the val of [NumExpr](#) expression

Parameters

<i>ot</i>	ostream
-----------	---------

Implements [Expr](#).

5.4.2.5 print()

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

the val of [NumExpr](#) expression

Parameters

<i>ot</i>	ostream
-----------	---------

Implements [Expr](#).

5.4.2.6 subst()

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

everywhere that the expression (whose subst method is called) contains a variable matching the string, the result Expr* should have the given replacement, instead.

Parameters

<i>s</i>	a std::string is substituted
<i>e</i>	a new expression

Returns

[Expr](#)

Implements [Expr](#).

5.4.3 Member Data Documentation

5.4.3.1 val

```
int NumExpr::val
```

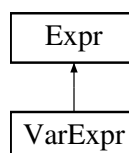
The documentation for this class was generated from the following files:

- /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/[expr.h](#)
- /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/[expr.cpp](#)

5.5 VarExpr Class Reference

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

Inheritance diagram for VarExpr:



Public Member Functions

- [VarExpr](#) (std::string *val*)
- bool [equals](#) ([Expr](#) *e)
- int [interp](#) ()
- bool [has_variable](#) ()
- [Expr](#) * [subst](#) (std::string s, [Expr](#) *e)
- void [print](#) (std::ostream &ot)
- void [pretty_print](#) (std::ostream &ot)

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 s, [Expr](#) *e)=0
- virtual void [print](#) (std::ostream &ot)=0
- std::string [to_string](#) ()
- virtual void [pretty_print](#) (std::ostream &ot)=0
- std::string [pretty_to_string](#) ()
- std::string [pretty_print_at](#) ([precedence_t](#) p)

Public Attributes

- std::string [val](#)

Additional Inherited Members

Public Types inherited from [Expr](#)

- enum [precedence_t](#) { [prec_none](#) , [prec_add](#) , [prec_mult](#) }

5.5.1 Constructor & Destructor Documentation

5.5.1.1 VarExpr()

```
VarExpr::VarExpr (
    std::string val )
```

constructor

Parameters

<i>val</i>	string value
------------	--------------

5.5.2 Member Function Documentation

5.5.2.1 equals()

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

if input [Expr](#) is a [VarExpr](#) and val equal

Parameters

<i>e</i>	input expression for comparing
----------	--------------------------------

Returns

true if *e* is [VarExpr](#) and *e*->val equals to this->val return true, otherwise false

Implements [Expr](#).

5.5.2.2 has_variable()

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

the var expression is a variable

Returns

true

Implements [Expr](#).

5.5.2.3 interp()

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

A variable has no value, so interp for a variable should throw a `std::runtime_error` exception

Exceptions

<i>std::runtime_error</i>	
---------------------------	--

Implements [Expr](#).

5.5.2.4 pretty_print()

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

the val of [VarExpr](#) expression

Parameters

<i>ot</i>	ostream
-----------	---------

Implements [Expr](#).

5.5.2.5 print()

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

[VarExpr](#) print: the val of [VarExpr](#) expression

Parameters

<i>ot</i>	ostream
-----------	---------

Implements [Expr](#).

5.5.2.6 subst()

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

everywhere that the expression (whose subst method is called) contains a variable matching the string, the result [Expr](#)* should have the given replacement, instead.

Parameters

<i>s</i>	a std::string is substituted
<i>e</i>	a new expression

Returns

[Expr](#)

Implements [Expr](#).

5.5.3 Member Data Documentation

5.5.3.1 val

```
std::string VarExpr::val
```

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

- [/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/expr.h](#)
- [/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/expr.cpp](#)

Chapter 6

File Documentation

6.1 /Users/zidongfan/myGithubRepo/CS6015_↔ 2023/assignment/cmdline.cpp File Reference

contains use_arguments function

```
#include "cmdline.h"
```

Functions

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

6.1.1 Detailed Description

contains use_arguments function

Author

Zidong Fan

Date

01/13/2023

6.1.2 Function Documentation

6.1.2.1 use_arguments()

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

6.2 /Users/zidongfan/myGithubRepo/CS6015_↵ 2023/assignment/cmdline.h File Reference

contains use_arguments function

```
#include <stdio.h>
#include <iostream>
#include <string>
```

Functions

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

6.2.1 Detailed Description

contains use_arguments function

Author

Zidong Fan

Date

01/13/2023

6.2.2 Function Documentation

6.2.2.1 use_arguments()

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

6.3 /Users/zidongfan/myGithubRepo/CS6015_↵ 2023/assignment/cmdline.h

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

```
00001
00009 #ifndef msdscript_h
00010 #define msdscript_h
00011
00012 #include <stdio.h>
00013 #include <iostream>
00014 #include <string>
00015
00016 void use_arguments(int argc, const char * argv[]);
00017
00018 #endif /* msdscript_h */
00019
00020
```

6.4 /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/expr.cpp File Reference

contains all constructions and methods in expr class

```
#include "expr.h"
```

6.4.1 Detailed Description

contains all constructions and methods in expr class

Author

Zidong Fan

Date

01/20/2023

6.5 /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/expr.h File Reference

contains [Expr](#) class contains [NumExpr](#), [AddExpr](#), [MultExpr](#), [VarExpr](#) subclasses

```
#include <stdio.h>
#include <string>
#include <stdexcept>
#include <sstream>
```

Classes

- class [Expr](#)
- class [NumExpr](#)
- class [AddExpr](#)
- class [MultExpr](#)
- class [VarExpr](#)

6.5.1 Detailed Description

contains [Expr](#) class contains [NumExpr](#), [AddExpr](#), [MultExpr](#), [VarExpr](#) subclasses

abstract grammar of expressions is: $\langle \text{expr} \rangle = \langle \text{number} \rangle \mid \langle \text{expr} \rangle + \langle \text{expr} \rangle \mid \langle \text{expr} \rangle * \langle \text{expr} \rangle \mid \langle \text{variable} \rangle$

Author

Zidong Fan

Date

01/20/2023

6.6 /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/expr.h

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

```

00001
00016 #ifndef expr_h
00017 #define expr_h
00018
00019 #include <stdio.h>
00020 #include <string>
00021 #include <stdexcept>
00022 #include <sstream>
00023
00024 class Expr {
00025 public:
00031     virtual bool equals(Expr *e) = 0;
00032
00037     virtual int interp() = 0;
00038
00043     virtual bool has_variable() = 0;
00044
00051     virtual Expr *subst(std::string s, Expr *e) = 0;
00052
00057     virtual void print(std::ostream &ot) = 0;
00058
00063     std::string to_string();
00064
00071     virtual void pretty_print(std::ostream &ot) = 0;
00072
00077     std::string pretty_to_string();
00078
00079
00080     //Hint 2: Here's the C++ way to declare a type precedence_t that has the possible values
    prec_none, prec_add, and prec_mult, in that order:
00081     typedef enum {
00082         prec_none,      // = 0
00083         prec_add,       // = 1
00084         prec_mult       // = 2
00085     } precedence_t;
00086
00087     //Hint: You will likely find it useful to "accumulate" a precedence level for parentheses as an
    argument to a pretty_print_at helper, where the precedence indicates which operations need
    parentheses: "none," "addition or lower," or "multiplication or lower."
00093     std::string pretty_print_at(precedence_t p);
00094 };
00095
00096 class NumExpr : public Expr {
00097 public:
00098     /* the value of NumExpr*/
00099     int val;
00100
00105     NumExpr(int val);
00106
00112     bool equals(Expr *e);
00113
00118     int interp();
00119
00124     bool has_variable();
00125
00132     Expr *subst(std::string s, Expr *e);
00133
00138     void print(std::ostream &ot);
00139
00144     void pretty_print(std::ostream &ot);
00145 };
00146
00147 class AddExpr : public Expr {
00148 public:
00149     /* left hand side expression of AddExpr */
00150     Expr *lhs;
00151     /* right hand side expression of AddExpr */
00152     Expr *rhs;
00153
00159     AddExpr(Expr *lhs, Expr *rhs);
00160
00166     bool equals(Expr *e);
00167
00172     int interp();
00173
00178     bool has_variable();
00179
00186     Expr *subst(std::string s, Expr *e);
00187
00192     void print(std::ostream &ot);
00193
00200     void pretty_print(std::ostream &ot);

```

```

00201 };
00202
00203 class MultExpr : public Expr {
00204 public:
00205     /* left hand side expression of MultExpr */
00206     Expr *lhs;
00207     /* right hand side expression of MultExpr */
00208     Expr *rhs;
00209
00215     MultExpr(Expr *lhs, Expr *rhs);
00216
00222     bool equals(Expr *e);
00223
00228     int interp();
00229
00234     bool has_variable();
00235
00243     Expr *subst(std::string s, Expr *e);
00244
00249     void print(std::ostream &ot);
00250
00257     void pretty_print(std::ostream &ot);
00258 };
00259
00260 class VarExpr : public Expr {
00261 public:
00262     /* the string value of VarExpr */
00263     std::string val;
00264
00269     VarExpr(std::string val);
00270
00276     bool equals(Expr *e);
00277
00278
00283     int interp();
00284
00289     bool has_variable();
00290
00297     Expr *subst(std::string s, Expr *e);
00298
00304     void print(std::ostream &ot);
00305
00310     void pretty_print(std::ostream &ot);
00311 };
00312
00313 #endif /* expr_h */

```

6.7 /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/main.cpp File Reference

int [main\(\)](#) to run the catch tests

```

#include <iostream>
#include "catch.h"

```

Macros

- `#define` [CATCH_CONFIG_RUNNER](#)

Functions

- int [main](#) (int argc, const char *argv[])

6.7.1 Detailed Description

int [main\(\)](#) to run the catch tests

6.7.2 Macro Definition Documentation

6.7.2.1 CATCH_CONFIG_RUNNER

```
#define CATCH_CONFIG_RUNNER
```

6.7.3 Function Documentation

6.7.3.1 main()

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

6.8 /Users/zidongfan/myGithubRepo/CS6015_2023/assignment/test↵ Expr.cpp File Reference

```
#include "expr.h"
#include "catch.h"
```

Functions

- [TEST_CASE](#) ("equals")
- [TEST_CASE](#) ("interp")
- [TEST_CASE](#) ("has_variable")
- [TEST_CASE](#) ("subst")
- [TEST_CASE](#) ("print")
- [TEST_CASE](#) ("pretty_print")

6.8.1 Function Documentation

6.8.1.1 TEST_CASE() [1/6]

```
TEST_CASE (
    "equals" )
```


6.8.1.2 TEST_CASE() [2/6]

```
TEST_CASE (
    "has_variable" )
```

6.8.1.3 TEST_CASE() [3/6]

```
TEST_CASE (
    "interp" )
```

interp Test Examples: CHECK((new MultExpr(new NumExpr(3), new NumExpr(2))) ->interp() == 6); CHECK((new AddExpr(new AddExpr(new NumExpr(10), new NumExpr(15)), new AddExpr(new NumExpr(20), new NumExpr(20))) ->interp() == 65);

6.8.1.4 TEST_CASE() [4/6]

```
TEST_CASE (
    "pretty_print" )
```

6.8.1.5 TEST_CASE() [5/6]

```
TEST_CASE (
    "print" )
```

6.8.1.6 TEST_CASE() [6/6]

```
TEST_CASE (
    "subst" )
```


Index

[/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/cmdline.cpp](#), 16
[29](#)

[/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/cmdline.h](#), 30

[/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/expr.cpp](#), 14
[31](#)

[/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/expr.h](#), 22
[31](#)

[/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/main.cpp](#), 33

[/Users/zidongfan/myGithubRepo/CS6015_2023/assignment/testExpr.cpp](#), 34

[AddExpr](#), 9

- [AddExpr](#), 10
- [equals](#), 10
- [has_variable](#), 11
- [interp](#), 11
- [lhs](#), 12
- [pretty_print](#), 11
- [print](#), 12
- [rhs](#), 12
- [subst](#), 12

[CATCH_CONFIG_RUNNER](#)

- [main.cpp](#), 34

[cmdline.cpp](#)

- [use_arguments](#), 29

[cmdline.h](#)

- [use_arguments](#), 30

[equals](#)

- [AddExpr](#), 10
- [Expr](#), 14
- [MultExpr](#), 18
- [NumExpr](#), 22
- [VarExpr](#), 26

[Expr](#), 13

- [equals](#), 14
- [has_variable](#), 14
- [interp](#), 14
- [prec_add](#), 14
- [prec_mult](#), 14
- [prec_none](#), 14
- [precedence_t](#), 13
- [pretty_print](#), 15
- [pretty_print_at](#), 15
- [pretty_to_string](#), 15
- [print](#), 16
- [subst](#), 16

[has_variable](#)

- [AddExpr](#), 11
- [Expr](#), 14
- [MultExpr](#), 19
- [NumExpr](#), 22
- [VarExpr](#), 26

[lhs](#)

- [AddExpr](#), 12
- [MultExpr](#), 20

[main](#)

- [main.cpp](#), 34

[main.cpp](#)

- [CATCH_CONFIG_RUNNER](#), 34
- [main](#), 34

[MultExpr](#), 17

- [equals](#), 18
- [has_variable](#), 19
- [interp](#), 19
- [lhs](#), 20
- [MultExpr](#), 18
- [pretty_print](#), 19
- [print](#), 20
- [rhs](#), 20
- [subst](#), 20

[NumExpr](#), 21

- [equals](#), 22
- [has_variable](#), 22
- [interp](#), 23
- [NumExpr](#), 22
- [pretty_print](#), 23
- [print](#), 23
- [subst](#), 24
- [val](#), 24

[prec_add](#)

- [Expr](#), 14

[prec_mult](#)

- [Expr](#), 14

[prec_none](#)

- Expr, [14](#)
- precedence_t
 - Expr, [13](#)
- pretty_print
 - AddExpr, [11](#)
 - Expr, [15](#)
 - MultExpr, [19](#)
 - NumExpr, [23](#)
 - VarExpr, [27](#)
- pretty_print_at
 - Expr, [15](#)
- pretty_to_string
 - Expr, [15](#)
- print
 - AddExpr, [12](#)
 - Expr, [16](#)
 - MultExpr, [20](#)
 - NumExpr, [23](#)
 - VarExpr, [27](#)
- rhs
 - AddExpr, [12](#)
 - MultExpr, [20](#)
- subst
 - AddExpr, [12](#)
 - Expr, [16](#)
 - MultExpr, [20](#)
 - NumExpr, [24](#)
 - VarExpr, [27](#)
- TEST_CASE
 - testExpr.cpp, [34](#), [35](#)
- testExpr.cpp
 - TEST_CASE, [34](#), [35](#)
- to_string
 - Expr, [16](#)
- use_arguments
 - cmdline.cpp, [29](#)
 - cmdline.h, [30](#)
- val
 - NumExpr, [24](#)
 - VarExpr, [28](#)
- VarExpr, [24](#)
 - equals, [26](#)
 - has_variable, [26](#)
 - interp, [26](#)
 - pretty_print, [27](#)
 - print, [27](#)
 - subst, [27](#)
 - val, [28](#)
 - VarExpr, [25](#)