Fraction Expression Solver

Generated by Doxygen 1.8.5

Wed Dec 11 2013 01:34:36

Contents

1 Class Index **1**

1.1 Class List 1

2 Class Documentation **3**

2.1 expressionParser Class Reference 3

2.2 expressionSolver Class Reference 3

2.3 frac Class Reference 3

Chapter 1

Class Index

1.1 Class List

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

expressionParser 3
expressionSolver 3
frac 3

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

/Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/expressionParser.cpp . . . . . ??
/Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/expressionParser.h . . . . . ??
/Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/expressionSolver.cpp . . . . . ??
/Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/expressionSolver.h . . . . . ??
/Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/frac.cpp . . . . . ??
/Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/frac.h . . . . . ??
/Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/Fraction.cpp . . . . . ??

Chapter 3

Class Documentation

3.1 expressionParser Class Reference

`#include <expressionParser.h>`

Public Member Functions

- expressionParser (std::string exp)
- virtual ~expressionParser ()
- std::string getPostFixedExpression ()

Private Types

- enum operatorType { NONE, UNARY, BINARY }

Private Member Functions

- void shuntingYard ()
- bool isOperator (char o)
- bool isOperand (char o)
- int getPrecedence (char o)

Private Attributes

- std::map< char, int > operatorPrecedenceTable
- enum operatorType expectedOperatorType
- std::string inFixExpression
- std::string postFixExpression

3.1.1 Member Enumeration Documentation

3.1.1.1 enum expressionParser::operatorType [private]

Enumerator

NONE

UNARY

BINARY

3.1.2 Constructor & Destructor Documentation

3.1.2.1 expressionParser::expressionParser (std::string *exp*)

Constructor Stores the expression to be Parsed as class variable and creates an Operator precedence table as a map.

3.1.2.2 expressionParser::~expressionParser ( ) [virtual]

Destructor

3.1.3 Member Function Documentation

3.1.3.1 std::string expressionParser::getPostFixedExpression ( )

Uses the infix Expression provided to the class to postFixed expression using Shunting Yard algorithm. Creates a postfixed Expression as a string.

3.1.3.2 int expressionParser::getPrecedence (char *o*) [private]

Gets precedence value of an operator

3.1.3.3 bool expressionParser::isOperand (char *o*) [private]

Checks if current operator is an operand

3.1.3.4 bool expressionParser::isOperator (char *o*) [private]

Checks of current token is an operator

3.1.3.5 void expressionParser::shuntingYard ( ) [private]

Shunting Yard algorithm for infix to postfix conversion of expression given a set of operators and their precedence, associativity. For current exercise I considered all operators as left associative.

3.1.4 Member Data Documentation

3.1.4.1 enum operatorType expressionParser::expectedOperatorType [private]

3.1.4.2 std::string expressionParser::inFixExpression [private]

3.1.4.3 std::map<char,int> expressionParser::operatorPrecedenceTable [private]

3.1.4.4 std::string expressionParser::postFixExpression [private]

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

- /Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/expressionParser.h
- /Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/expressionParser.cpp

3.2 expressionSolver Class Reference

`#include <expressionSolver.h>`

Public Member Functions

- expressionSolver (std::string expr)
- virtual ~expressionSolver ()
- frac * solveExpression ()

Private Member Functions

- bool isOperator (std::string o)
- void useOperator (std::string o, std::stack< frac * > &postFixExprStack)
- void pushOperandToStack (std::string o, std::stack< frac * > &postFixExprStack)

Private Attributes

- std::string postFixedExpression

3.2.1 Constructor & Destructor Documentation

3.2.1.1 expressionSolver::expressionSolver (std::string *expr*)

Constructor

3.2.1.2 expressionSolver::~expressionSolver ( ) [virtual]

Destructor

3.2.2 Member Function Documentation

3.2.2.1 bool expressionSolver::isOperator (std::string *o*) [private]

Checks if current token is an operator

3.2.2.2 void expressionSolver::pushOperandToStack (std::string *currToken,* std::stack< frac * > & *postFixExprStack*) [private]

Push Operand to Stack. This function converts each integer obtained to a fraction class object. This facilitates use of operators on operands without checking if its a fraction or an integer.

3.2.2.3 frac * expressionSolver::solveExpression ( )

Solves a Post Fix Expression. It parses the Postfix expression string, creates objects for operands and uses a stack to keep track of results.

3.2.2.4 void expressionSolver::useOperator (std::string *currToken,* std::stack< frac * > & *postFixExprStack*) [private]

Uses Operator If current token parsed is an operator, pulls required number of operands from stack and operates on them.

3.2.3 Member Data Documentation

3.2.3.1 std::string expressionSolver::postFixedExpression [private]

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

- /Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/expressionSolver.h
- /Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/expressionSolver.cpp

3.3 frac Class Reference

`#include <frac.h>`

Public Member Functions

- frac (int n, int d)
- frac (frac &f)
- frac (frac *f)
- ~frac ()
- void display () const
- int getDinominator ()
- int getNumerator ()
- void reduceFraction ()
- frac * clone ()
- frac & operator+ (frac &f)
- frac & operator- (frac &f)
- frac & operator* (frac &f)
- frac & operator/ (frac &f)
- frac & operator- ()

  unary - operator overload

Private Member Functions

- int getLcm (int dinominator1, int dinominator2)

  Get LCM.
- int getGcd (int a, int b)

  Get GCD.

Private Attributes

- int numerator
- int dinominator

3.3.1 Constructor & Destructor Documentation

3.3.1.1 frac::frac (int *n,* int *d*)

Constructs Fraction

The function creates the fraction object and stores it in its reduced form.

3.3.1.2 frac::frac (frac & *f*)

Copy Constructor: Copies and creates a frac object using reference to frac object

3.3.1.3 frac::frac (frac * *f*)

Copy Constructor: Copies and creates a frac object using pointer to another frac object

3.3.1.4 frac::~frac ( )

Destructor

3.3.2 Member Function Documentation

3.3.2.1 frac * frac::clone ( )

Clones an object

3.3.2.2 void frac::display ( ) const

The function displays the fraction in proper form. An improper fraction is converted to mixed form

3.3.2.3 int frac::getDinominator ( )

Retrieves Dinominator

3.3.2.4 int frac::getGcd (int *a,* int *b*) [private]

Get GCD.

3.3.2.5 int frac::getLcm (int *a,* int *b*) [private]

Get LCM.

3.3.2.6 int frac::getNumerator ( )

Retrieves Numerator

3.3.2.7 frac & frac::operator* (frac & *secondOperand*)

This function multiplies two fractions and modifies the first operand to the result

3.3.2.8 frac & frac::operator+ (frac & *secondOperand*)

This function adds two fractions and modifies the first operand to the result

3.3.2.9 frac & frac::operator- (frac & *secondOperand*)

This function subtracts second fraction from first and modifies the first operand to the result

3.3.2.10 frac & frac::operator- ( )

unary - operator overload

This function negates the fraction (argument)

3.3.2.11 frac & frac::operator/ (frac & *secondOperand*)

This function divides first fraction by second and modifies the first operand to the result

3.3.2.12 void frac::reduceFraction ( )

The function reduces a fraction to its lowest form by dividing both numerator and dinominator by their GCD.

3.3.3 Member Data Documentation

3.3.3.1 int frac::dinominator [private]

3.3.3.2 int frac::numerator [private]

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

- /Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/frac.h
- /Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/frac.cpp

Chapter 4

File Documentation

4.1 /Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/expressionParser.cpp File Reference

```
#include "expressionParser.h"
#include <sstream>
#include <iostream>
#include <stack>
```

Include dependency graph for expressionParser.cpp:

4.2 /Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/expressionParser.h File Reference

```
#include <string>
#include <map>
#include "frac.h"
```

Include dependency graph for expressionParser.h: This graph shows which files directly or indirectly include this file:

Classes

- class expressionParser

4.3 /Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/expressionSolver.cpp File Reference

```
#include "expressionSolver.h"
#include <stack>
#include <iostream>
#include <vector>
#include <sstream>
#include "frac.h"
```

Include dependency graph for expressionSolver.cpp:

4.4 /Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/expressionSolver.h File Reference

```
#include <string>
#include <stack>
```

Include dependency graph for expressionSolver.h: This graph shows which files directly or indirectly include this file:

Classes

- class expressionSolver

4.5 /Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/frac.cpp File Reference

```
#include "frac.h"
#include <iostream>
#include <cmath>
```

Include dependency graph for frac.cpp:

4.6 /Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/frac.h File Reference

This graph shows which files directly or indirectly include this file:

Classes

- class frac

4.7 /Users/phanindrabhagavatula/Everything Useful/eclipse/Fraction/src/Fraction.cpp File Reference

```
#include <iostream>
#include <string>
#include "frac.h"
#include "expressionParser.h"
#include "expressionSolver.h"
```

Include dependency graph for Fraction.cpp:

Functions

- void terminator ()
- int main ()

4.7.1 Function Documentation

4.7.1.1 int main ( )

4.7.1.2 void terminator ( )