Expression Tree

Generated by Doxygen 1.7.6.1

Tue Mar 4 2014 22:44:44

Contents

1 Class Index **1**

1.1 Class List 1

2 File Index **3**

2.1 File List 3

3 Class Documentation **5**

3.1 ExprTree< DataType > Class Template Reference 5

3.1.1 Constructor & Destructor Documentation 5

3.1.1.1 ExprTree 5

3.1.1.2 ExprTree 6

3.1.1.3 ~ExprTree 6

3.1.2 Member Function Documentation 6

3.1.2.1 build 6

3.1.2.2 clear 7

3.1.2.3 commute 7

3.1.2.4 evaluate 7

3.1.2.5 expression 8

3.1.2.6 isEquivalent 8

3.1.2.7 operator= 8

4 File Documentation **11**

4.1 ExpressionTree.cpp File Reference 11

4.1.1 Detailed Description 11

Chapter 1

Class Index

1.1 Class List

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

ExprTree< DataType > 5

Chapter 2

File Index

2.1 File List

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

config.h . . . . . **??**
ExpressionTree.cpp
This program creates and expression tree and evaluates it 11
ExpressionTree.h . . . . . **??**

Chapter 3

Class Documentation

3.1 ExprTree< DataType > Class Template Reference

Classes

- class **ExprTreeNode**

Public Member Functions

- ExprTree ()
- ExprTree (const ExprTree &source)
- ExprTree & operator= (const ExprTree &source)
- ~ExprTree ()
- void build ()
- void expression () const
- DataType evaluate () const throw (logic_error)
- void clear ()
- void commute ()
- bool isEquivalent (const ExprTree &source) const
- void **showStructure** () const

template<typename DataType> class ExprTree< DataType >

3.1.1 Constructor & Destructor Documentation

3.1.1.1 template<typename DataType > ExprTree< DataType >::ExprTree ( )

Default constructor.

The default constructor initializes the root to null.

Postcondition

root is equal to null.

3.1.1.2 template<typename DataType > ExprTree< DataType >::ExprTree (const ExprTree< DataType > & *source*)

Copy constructor.

The copy constructor accepts a tree and creates a tree. It checks to see if the source tree is empty. If it is empty then it sets the root equal to null otherwise it copies over the source tree

Precondition

a tree does not exist.

Postcondition

a tree is copied from the source tree.

Parameters

ExprTree	source.

3.1.1.3 template<typename DataType > ExprTree< DataType >::~ExprTree ( )

Destructor.

The destuctor deallocates memory if the tree is not empty. It does this by calling the clear function.

Precondition

The tree has nodes or is empty.

Postcondition

If the tree had nodes then they are deallocated.

3.1.2 Member Function Documentation

3.1.2.1 template<typename DataType > void ExprTree< DataType >::build ( )

Build.

The build function builds the tree. It does this by calling the build helper function.

Precondition

The tree is empty.

Postcondition

The tree is built.

3.1.2.2 template<typename DataType > void ExprTree< DataType >::clear ( )

Clear.

The clear function deallocates any memory that was used for a tree. It does this by calling clear helper.

Precondition

A tree will have memory and root will contain a value.

Postcondition

Any memory a tree had will be deallocated and root is set to null.

3.1.2.3 template<typename DataType > void ExprTree< DataType >::commute ( )

Commute.

This funciton applies the commutative property to the tree. The function checks to see if the tree is empty. If its not empty then it calls commuteHelper. It clears the dest tree and uses copy helper to create the new commuted tree.

Precondition

A tree must exist.

Postcondition

A tree is recreated with the commutive properties applied.

3.1.2.4 template<typename DataType > DataType ExprTree< DataType >::evaluate ( ) const throw (logic_error)

Evalutate.

This function evalutes the tree and returns the DataType. It does this by calling the evalHelper function.

Precondition

A tree will note be evaluated.

Postcondition

The tree will be evaluated.

Returns

DataType

3.1.2.5 template<typename DataType > void ExprTree< DataType >::expression ( ) const

Expression.

This function outputs the tree in an expression format. It checks to see if the tree is not empty. It it contains data then it checks to see if the root is a digit. If it is then it outputs the digit. Otherwise it calls expressionHelper to express the expression.

Precondition

A tree will not be expressed.

Postcondition

A tree will be expressed.

3.1.2.6 template<typename DataType > bool ExprTree< DataType >::isEquivalent (const ExprTree< DataType > & *source*) const

isEquvialent.

This function checks to see if two trees are equal. It does this by calling equalHelper.

Returns

Returns a bool with the result if it is equal or not.

3.1.2.7 template<typename DataType > ExprTree< DataType > & ExprTree< DataType >::operator= (const ExprTree< DataType > & *source*)

Overloaded assignment operator.

The assignment operator copies the source tree over. It accepts a source tree and checks to see if 'this' is equal to source. If it is not equal to the source then it checks to see if the source tree is empty. If it is empty then the tree is cleared and root is set equal to null. Otherwise the contents of the source tree is copied over.

Precondition

The tree exists with current data.

Postcondition

The tree is assigned the contents of a source tree.

Parameters

ExprTree	source

Returns

∗this

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

- ExpressionTree.h
- ExpressionTree.cpp
- ExpressionTreeWIP.cpp
- show8.cpp

Chapter 4

File Documentation

4.1 ExpressionTree.cpp File Reference

This program creates and expression tree and evaluates it.

`#include <ctype.h> #include <stdlib.h> #include "Expression-Tree.h" #include "show8.cpp"`

4.1.1 Detailed Description

This program creates and expression tree and evaluates it.

Author

Amardeep Singh

Date

March 4th 2014