

## Lab 8 - Expression Trees

Generated by Doxygen 1.8.5

Thu Oct 24 2013 00:35:25



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
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 . . . . .	6
3.1.2.3	commute . . . . .	7
3.1.2.4	evaluate . . . . .	7
3.1.2.5	expression . . . . .	7
3.1.2.6	isEquivalent . . . . .	7
3.1.2.7	operator= . . . . .	8
3.1.2.8	showStructure . . . . .	8
<b>4</b>	<b>File Documentation</b>	<b>9</b>
4.1	config.h File Reference . . . . .	9
4.1.1	Macro Definition Documentation . . . . .	9
4.1.1.1	LAB8_TEST1 . . . . .	9
4.1.1.2	LAB8_TEST2 . . . . .	9
4.1.1.3	LAB8_TEST3 . . . . .	9
4.2	ExpressionTree.cpp File Reference . . . . .	9
4.3	ExpressionTree.h File Reference . . . . .	9
4.4	show8.cpp File Reference . . . . .	10
4.5	test8.cpp File Reference . . . . .	10

4.5.1	Function Documentation . . . . .	10
4.5.1.1	dummy . . . . .	10
4.5.1.2	main . . . . .	10
 <b>Index</b>		 <b>11</b>

# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">ExprTree&lt; DataType &gt;</a> . . . . .	5
--	---



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">config.h</a>	9
<a href="#">ExpressionTree.cpp</a>	9
<a href="#">ExpressionTree.h</a>	9
<a href="#">show8.cpp</a>	10
<a href="#">test8.cpp</a>	10





## Chapter 3

# Class Documentation

### 3.1 ExprTree< DataType > Class Template Reference

```
#include <ExpressionTree.h>
```

#### 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

#### 3.1.1 Constructor & Destructor Documentation

##### 3.1.1.1 `template<typename > ExprTree< typename >::ExprTree ( )`

#### Precondition

New [ExprTree](#) Class

#### Postcondition

root pointing to null

#### Algorithm:

- point root to null

#### Exceptional/Error Conditions:

- none

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

**Precondition**

new [ExprTree](#) class

**Postcondition**

a deep copy of source [ExprTree](#)

**Parameters**

<i>source</i>	Source <a href="#">ExprTree</a> to be deep copied
---------------	---

**Algorithm:**

- Use overloaded assignment operator to copy the data from the source to this class

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

**Precondition**

an ExprTreeClass

**Postcondition**

deallocated ExprTreeClass

**Algorithm:**

- Use clear member function to deallocate the tree

## 3.1.2 Member Function Documentation

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

**Precondition**

an ExprTreeClass

**Postcondition**

a built Expression tree

**Algorithm:**

- Use recBuild member function to make the expression tree

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

**Precondition**

an ExprTreeClass

**Postcondition**

deallocated ExprTreeClass

**Algorithm:**

- use the recClear member function to deallocate the exprTree

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

an ExprTreeClass

**Postcondition**

commuted Expression Tree

**Algorithm:**

- use the recCommute member function to commute the exprTree

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

a built ExprTreeClass

**Postcondition**

The mathematical expression is evaluated

**Returns**

The mathematical evaluation of the expression of the tree

**Algorithm:**

- use the recEvaluate member function to evaluate the exprTree

**Exceptions**

<i>A</i>	logic error is thrown if the tree is empty
----------	--

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

a built ExprTreeClass

**Postcondition**

The mathematical expression is printed on screen infix notation

**Algorithm:**

- Use recExpression member function to print the expression

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

an ExprTreeClass

**Postcondition**

returned boolean if the trees are equivalent

**Parameters**

<i>source</i>	Source <a href="#">ExprTree</a> to be compared
---------------	--

**Returns**

This function returns if the trees are equivalent based on commutative properties

**Algorithm:**

- use the `recIsEquivalent` member function to check equivalence

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

**Precondition**

an `ExprTreeClass`

**Postcondition**

a deep copy of source [ExprTree](#)

**Parameters**

<i>source</i>	Source <a href="#">ExprTree</a> to be deep copied
---------------	---

**Algorithm:**

- Use `recCopy` to make a copy of each node in the [ExprTree](#)

**3.1.2.8** `template<typename DataType > void ExprTree< DataType >::showStructure ( ) const`

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

- [ExpressionTree.h](#)
- [ExpressionTree.cpp](#)
- [show8.cpp](#)

# Chapter 4

## File Documentation

### 4.1 config.h File Reference

#### Macros

- `#define LAB8_TEST1 0`
- `#define LAB8_TEST2 1`
- `#define LAB8_TEST3 1`

#### 4.1.1 Macro Definition Documentation

##### 4.1.1.1 `#define LAB8_TEST1 0`

Expression Tree class (Lab 8) configuration file. Activate test #N by defining the corresponding LAB8\_TESTN to have the value 1.

##### 4.1.1.2 `#define LAB8_TEST2 1`

##### 4.1.1.3 `#define LAB8_TEST3 1`

### 4.2 ExpressionTree.cpp File Reference

```
#include <iostream>
#include "ExpressionTree.h"
```

### 4.3 ExpressionTree.h File Reference

```
#include <stdexcept>
#include <iostream>
```

#### Classes

- class `ExprTree< DataType >`

## 4.4 show8.cpp File Reference

## 4.5 test8.cpp File Reference

```
#include <iostream>
#include <stdexcept>
#include "ExpressionTree.cpp"
#include "config.h"
```

### Functions

- `template<typename DataType > void dummy (ExprTree< DataType > copyTree)`
- `int main ()`

### 4.5.1 Function Documentation

4.5.1.1 `template<typename DataType > void dummy ( ExprTree< DataType > copyTree )`

4.5.1.2 `int main ( )`

# Index

- ~ExprTree
  - ExprTree, [6](#)
- build
  - ExprTree, [6](#)
- clear
  - ExprTree, [6](#)
- commute
  - ExprTree, [6](#)
- config.h, [9](#)
  - LAB8\_TEST1, [9](#)
  - LAB8\_TEST2, [9](#)
  - LAB8\_TEST3, [9](#)
- dummy
  - test8.cpp, [10](#)
- evaluate
  - ExprTree, [7](#)
- ExprTree
  - ~ExprTree, [6](#)
  - build, [6](#)
  - clear, [6](#)
  - commute, [6](#)
  - evaluate, [7](#)
  - ExprTree, [5](#)
  - expression, [7](#)
  - ExprTree, [5](#)
  - isEquivalent, [7](#)
  - operator=, [8](#)
  - showStructure, [8](#)
- ExprTree< DataType >, [5](#)
- expression
  - ExprTree, [7](#)
- ExpressionTree.cpp, [9](#)
- ExpressionTree.h, [9](#)
- isEquivalent
  - ExprTree, [7](#)
- LAB8\_TEST1
  - config.h, [9](#)
- LAB8\_TEST2
  - config.h, [9](#)
- LAB8\_TEST3
  - config.h, [9](#)
- main
  - test8.cpp, [10](#)
- operator=
  - ExprTree, [8](#)
- show8.cpp, [10](#)
- showStructure
  - ExprTree, [8](#)
- test8.cpp, [10](#)
  - dummy, [10](#)
  - main, [10](#)