MathExpressionSolverLib

Generated by Doxygen 1.8.18

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

Data Structure Documentation

2.1 ExpressionTree Struct Reference

Public Member Functions

- int get_lower_precedence (ExpressionTree ∗etree, char ∗expr, int start, int end)
  Encontra o caractere com menor precedencia a partir de uma cadeia de caracteres.
- int precedence (char c)
  Atribui um valor de precedencia a partir do caractere.
- int parse (ExpressionTree ∗etree, char ∗expr, int start, int end)
  Função recursiva para fazer o parse da expressão.
- ExpressionTree ∗ parse_expr (char ∗expr, int size)
  Cria e popula uma arvore com a expressão fornecida.
- ExpressionTree ∗ init_etree ()
  Inicializa a arvore.
- float evaluate (ExpressionTree ∗etree)
  Faz a resolução da expressão.
- int save_as_dot (ExpressionTree ∗etree, char ∗filepath)
  Salva a arvore em um formato para geração do grafo.
- void free_mem_expressiontree (ExpressionTree ∗etree)
  Libera a memoria da ExpressionTree.

Data Fields

- char value
  Valor do nó
- struct ExpressionTree ∗ left
  Nós filhos.
- struct ExpressionTree ∗ **right**

2.1.1 Member Function Documentation

2.1.1.1 evaluate()

```
float evaluate (
            ExpressionTree * etree )
```

Faz a resolução da expressão.

Parameters

etree	ExpressionTree vazia

Returns

retorna o ExpressionTree equivalente a expressão.

2.1.1.2 free_mem_expressiontree()

```
void free_mem_expressiontree (
            ExpressionTree * etree )
```

Libera a memoria da ExpressionTree.

Parameters

etree	Arvore binaria a ser liberada

2.1.1.3 get_lower_precedence()

```
int get_lower_precedence (
            ExpressionTree * etree,
            char * expr,
            int start,
            int end )
```

Encontra o caractere com menor precedencia a partir de uma cadeia de caracteres.

Returns

Retorna ao item de menor precedencia.

2.1.1.4 init_etree()

```
ExpressionTree * init_etree ( )
```

Inicializa a arvore.

Returns

Retorna a uma arvore sem nós

2.1.1.5 parse()

```
int parse (
            ExpressionTree * etree,
            char * expr,
            int start,
            int end )
```

Função recursiva para fazer o parse da expressão.

Parameters

etree	Arvore binaria a ser populada

Returns

Retorna 1 se não existir problemas.

2.1.1.6 parse_expr()

```
ExpressionTree * parse_expr (
            char * expr,
            int size )
```

Cria e popula uma arvore com a expressão fornecida.

Parameters

expr	Uma string contendo a expressão sem espaços
size	Tamanho da string

Returns

retorna a arvore criada a partir da expressão

2.1.1.7 precedence()

```
int precedence (
            char c )
```

Atribui um valor de precedencia a partir do caractere.

Parameters

c	Caractere a ser avaliado

Returns

retorna um valor de 1 a 3 como a precedencia.

2.1.1.8 save_as_dot()

```
int save_as_dot (
            ExpressionTree * etree,
            char * filepath )
```

Salva a arvore em um formato para geração do grafo.

Parameters

etree	A arvore de expressão
filepath	Local onde será salvo o arquivo

Returns

1 se ok 0 se não ok

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

- MathExpressionSolverLib/expression_tree.c
- MathExpressionSolverLib/expression_tree.h

Index