functions.h File Reference

```
#include <stack>
#include <sstream>
#include <map>
#include <iterator>
```

Go to the source code of this file.

Classes

struct **Entry_**
Struct for storing operators and expressions. More...

Functions

bool **PrecedenceLess** (const string &left, const string &right, bool assoc)
Function for determining precedence of operations More...

void **Parenthesize** (**Entry_** *old, const string &token, bool assoc)
Function to parenthesize low-precedence operations More...

void **AddToken** (stack< **Entry_** > *stack, const string &token)
Function to add operation sign to the expression More...

string **ToInfix** (const string &src)
Function to parse postfix notation into infix notation More...

template<typename ITERATOR >
double **calcRPN** (ITERATOR iter, ITERATOR end)
Function to calculate postfix expression More...

bool **IsOperator** (string C)
Function to verify whether a character/word is operator symbol or not. More...

bool **IsOpeningParenthesis** (string C)
Function to verify whether a character is an opening parenthesis More...

bool **IsClosingParenthesis** (string C)
Function to verify whether a character is a closing parenthesis More...

bool **IsRightAssociative** (string op)
Function to verify whether an operator is right associative or not. More...

int **GetOperatorWeight** (string op)
Function to get weight of an operator. An operator with higher weight will have higher precedence. More...

bool **HasHigherPrecedence** (string op1, string op2)
Function to determine precedence between two operators More...

template<typename ITERATOR >
string **ToPostfix** (ITERATOR iter, ITERATOR end)
Function to parse infix notation into postfix notation More...

template<typename ITERATOR >
bool **isRPN** (ITERATOR iter, ITERATOR end)
Function to check notation. True if postfix, false if infix More...

template<typename ITERATOR >
bool **isCorrect** (ITERATOR iter, ITERATOR end)
Function to check if the input is correct More...

Function Documentation

◆ AddToken()

```
void AddToken ( stack< Entry_ > *  stack,
                const string &     token
              )
```

Function to add operation sign to the expression

Parameters

stack	stack that holds items of **Entry_** struct
token	operation sign

◆ calcRPN()

```
template<typename ITERATOR >
double calcRPN ( ITERATOR  iter,
                 ITERATOR  end
               )
```

Function to calculate postfix expression

Template Parameters

ITERATOR

Parameters

iter	iterator to sweep through expression
end	end of the expression

Returns

result of the expression

◆ GetOperatorWeight()

```
int GetOperatorWeight ( string  op )
```

Function to get weight of an operator. An operator with higher weight will have higher precedence.

Parameters

op operator

Returns

operator's weight

◆ HasHigherPrecedence()

```
bool HasHigherPrecedence ( string  op1,
                           string  op2
                         )
```

Function to determine precedence between two operators

Parameters

op1 first operator

op2 second operator

◆ IsClosingParenthesis()

```
bool IsClosingParenthesis ( string  C )
```

Function to verify whether a character is a closing parenthesis

Parameters

C

Returns

true if it's a closing parenthesis

◆ isCorrect()

```
template<typename ITERATOR >
bool isCorrect ( ITERATOR  iter,
                 ITERATOR  end
               )
```

Function to check if the input is correct

Template Parameters

ITERATOR

Parameters

iter	iterator to sweep through expression
end	end of the expression

Returns

true if expression is correct

◆ IsOpeningParenthesis()

```
bool IsOpeningParenthesis ( string  C )
```

Function to verify whether a character is an opening parenthesis

Parameters

C	a character

Returns

true if it's an opening parenthesis

◆ IsOperator()

```
bool IsOperator ( string  C )
```

Function to verify whether a character/word is operator symbol or not.

Parameters

C	a character/word

Returns

true if it's an operator

◆ IsRightAssociative()

```
bool IsRightAssociative ( string  op )
```

Function to verify whether an operator is right associative or not.

right-associative means that the operations are grouped from the right.

Parameters

op	operator

◆ isRPN()

```
template<typename ITERATOR >
bool isRPN ( ITERATOR  iter,
             ITERATOR  end
           )
```

Function to check notation. True if postfix, false if infix

Template Parameters

ITERATOR

Parameters

iter	iterator to sweep through expression
end	end of the expression

Returns

true if expression is in postfix, false if it's in infix

◆ Parenthesize()

```
void Parenthesize ( Entry_ *         old,
                    const string &  token,
                    bool            assoc
                  )
```

Function to parenthesize low-precedence operations

Parameters

old	old expression without parenthesizes
token	operation sign
assoc	a bool variable to determine if the operation is associative

◆ PrecedenceLess()

```
bool PrecedenceLess ( const string &  left,
                      const string &  right,
                      bool            assoc
                    )
```

Function for determining precedence of operations

Parameters

left	list of operators
right	list of precedence
assoc	a bool variable to determine if the operation is associative

◆ ToInfix()

```
string ToInfix ( const string &  src )
```

Function to parse postfix notation into infix notation

Parameters

src	expression to parse

Returns

parsed expression

◆ ToPostfix()

```
template<typename ITERATOR >
string ToPostfix ( ITERATOR  iter,
                   ITERATOR  end
                 )
```

Function to parse infix notation into postfix notation

Template Parameters

ITERATOR

Parameters

iter	iterator to sweep through expression
end	end of the expression

Returns

parsed expression