

五则运算计算器

制作者 Doxygen 1.9.1

1 类索引	1
1.1 类列表	1
2 文件索引	3
2.1 文件列表	3
3 类说明	5
3.1 ExpTree类 参考	5
3.1.1 详细描述	5
3.1.2 成员函数说明	5
3.1.2.1 getResult()	5
3.2 Node类 参考	6
3.2.1 详细描述	7
3.2.2 成员函数说明	7
3.2.2.1 calculateResult()	7
4 文件说明	9
4.1 calculator.h 文件参考	9
4.1.1 详细描述	10
4.1.2 宏定义说明	10
4.1.2.1 _EPS	10
4.1.3 函数说明	11
4.1.3.1 checkBrackets()	11
4.1.3.2 isDigit()	11
4.1.3.3 isInteger()	11
4.1.3.4 quickPow()	12
4.1.3.5 realEqual()	12
Index	13

Chapter 1

类索引

1.1 类列表

这里列出了所有类、结构、联合以及接口定义等，并附带简要说明:

ExpTree	表达式树类	5
Node	表达式树中的结点类	6

Chapter 2

文件索引

2.1 文件列表

这里列出了所有文档化的文件，并附带简要说明:

calculator.h	
2022 ~ 2023 秋冬 数据结构与算法 项目作业	9

Chapter 3

类说明

3.1 ExpTree类 参考

表达式树类

```
#include <calculator.h>
```

Public 成员函数

- bool `isEmpty` ()
判断树是否为空
- bool `getResult` (std ::string str, double *ret)
将中缀表达式存储到表达式树中

3.1.1 详细描述

表达式树类

3.1.2 成员函数说明

3.1.2.1 getResult()

```
bool ExpTree::getResult (  
    std ::string str,  
    double * ret ) [inline]
```

将中缀表达式存储到表达式树中

参数

str	表达式字符串
ret	表达式的计算结果；若出错则置为 -1

返回

返回表达式是（1）否（0）计算成功

该类的文档由以下文件生成:

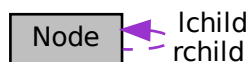
- [calculator.h](#)

3.2 Node类 参考

表达式树中的结点类

```
#include <calculator.h>
```

Node 的协作图:



Public 成员函数

- bool [calculateResult](#) ()
计算以当前结点为根的子树代表的表达式结果

Public 属性

- int [type](#)
存放结点类型（1——数字，2——符号）
- double [val](#)
结点数据域（结点中的数字，或符号：1——加，2——减，3——乘，4——除，5——乘方）
- [Node](#) * [lchild](#)
- [Node](#) * [rchild](#)
- double [res](#)
以该结点为根的子树的结果值

3.2.1 详细描述

表达式树中的结点类

3.2.2 成员函数说明

3.2.2.1 calculateResult()

```
bool Node::calculateResult ( ) [inline]
```

计算以当前结点为根的子树代表的表达式结果

返回

返回是 (1) 否 (0) 计算成功

该类的文档由以下文件生成:

- [calculator.h](#)

Chapter 4

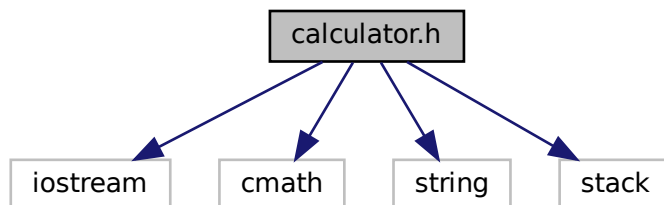
文件说明

4.1 calculator.h 文件参考

2022 ~ 2023 秋冬 数据结构与算法 项目作业

```
#include <iostream>
#include <cmath>
#include <string>
#include <stack>
```

calculator.h 的引用(Include)关系图:



类

- class `Node`
表达式树中的结点类
- class `ExpTree`
表达式树类

宏定义

- `#define EPS 1e-6`

函数

- `bool realEqual (const double a, const double b)`
判断两个双精度浮点数是否“相等”
- `bool isInteger (const double x)`
判断一个浮点数是否为整数
- `bool isLegal (std::string str)`
判断表达式子串 `str` 是否为合法子串
- `bool isDigit (std::string str)`
判断表达式中的合法子串是否为数字
- `double quickPow (const double a, const int b)`
利用快速幂计算 a^b
- `bool checkBrackets (std::string str)`
检测表达式括号是否匹配

4.1.1 详细描述

2022 ~ 2023 秋冬 数据结构与算法 项目作业

作者

王笑同 (3210105450@zju.edu.cn)

版本

1.3

日期

2022-11-05

版权所有

Copyright (c) 2022

4.1.2 宏定义说明

4.1.2.1 _EPS

```
#define _EPS 1e-6
```

待添加功能：基本初等函数、识别表达式错误（优化）、支持常数 e 和 π (pi) 例子：输入 1: $2^{(1+3)-5*(15.23)/(1+2)*3-5}$ 输出 1: -65.15 输入 2: $1.25 + [3*(1+2^2)*3-43](4-2)$ 输出 2: 5.25 输入 3: $0.2*((-5)^2)$ 输出 3: 5 输入 4: 3^{-1} 输出 4: 错误：表达式不合法！输入 5: $3^{(-1)}$ 输出 5: 0.333333 输入 6: (空字符串) 输出 6: (无) 输入 7: $+3.3*5+(+2-1)^2$ 输出 7: 16.5 输入 8: $3-(2.5-15/6)^{(8-8)}$ 输出 8: 错误：0^0 无意义！输入 9: $(-2)^3$ 输出 9: -8 输入 10: $(-2)^{3.000001}$ 输出 10: 错误：不能计算负数的非整数次幂！输入 11: $+2(1-3^{(5.6-4.6)})$ 输出 11: -4 输入 12: $(1+\{25/5\})^{([1.09]+0.91)}$ 输出 12: 错误：表达式括号不匹配！输入 13: 2^3^2 输出 13: 512 输入 14: $7/1-6+(0)-((9-7.5)^{(1.46*(-4)/5.02)^{(-5)+1.3})}$ 输出 14: -1.12672 输入 15: $6/4/(2^3.94/(5/((-5.87)/4)))+(-9)^2$ 输出 15: 80.9792

4.1.3 函数说明

4.1.3.1 checkBrackets()

```
bool checkBrackets (
    std ::string str )
```

检测表达式括号是否匹配

参数

str	待检测的表达式串
-----	----------

返回

返回匹配结果

4.1.3.2 isDigit()

```
bool isDigit (
    std ::string str )
```

判断表达式中的合法子串是否为数字

参数

str	待判断的字符串
-----	---------

返回

返回是 (1) 否 (0) 为数字

4.1.3.3 isInteger()

```
bool isInteger (
    const double x )
```

判断一个浮点数是否为整数

参数

x	待判断的浮点数
---	---------

返回

返回是否为浮点数

4.1.3.4 quickPow()

```
double quickPow (
    const double a,
    const int b )
```

利用快速幂计算 a^b

参数

a	底数（实数）
b	指数（整数）

返回

返回 a^b 的结果

4.1.3.5 realEqual()

```
bool realEqual (
    const double a,
    const double b )
```

判断两个双精度浮点数是否“相等”

参数

a	浮点数 1
b	浮点数 2

返回

返回在误差范围内是否相等

Index

- `_EPS`
 - `calculator.h`, [10](#)
- `calculateResult`
 - `Node`, [7](#)
- `calculator.h`, [9](#)
 - `_EPS`, [10](#)
 - `checkBrackets`, [11](#)
 - `isDigit`, [11](#)
 - `isInteger`, [11](#)
 - `quickPow`, [12](#)
 - `realEqual`, [12](#)
- `checkBrackets`
 - `calculator.h`, [11](#)
- `ExpTree`, [5](#)
 - `getResult`, [5](#)
- `getResult`
 - `ExpTree`, [5](#)
- `isDigit`
 - `calculator.h`, [11](#)
- `isInteger`
 - `calculator.h`, [11](#)
- `Node`, [6](#)
 - `calculateResult`, [7](#)
- `quickPow`
 - `calculator.h`, [12](#)
- `realEqual`
 - `calculator.h`, [12](#)