# 人工智能微专业

控制台计算器


Tongji University
School of Software Artificial Intelligence

**ID:** 1754060

**Name:** 张喆

**Academy:** 软件学院

**Course:** 数据结构

**Email:** dbzdbz@tongji.edu.cn

# 控制台计算器(Cherry-Calculator)

### Table of Contents



---

## 功能介绍

- [x] 计算表达式的值（支持括号）：`math [expression]`
- [x] 对表达式进行美化: `math -beauty [expression]`
- [x] 将中缀表达式转换为后缀表达式: `math -rev [expression]`
- [x] 检测表达式是否有非法字符
- [x] 检测表达式括号是否匹配
- [x] 查看帮助信息: `help`
- [x] 提示未找到该指令

## 开发环境

- **操作系统**
  - **开发环境**：macOS Catalina 10.15.4
  - **打包环境**：Windows 10
- **IDE**
  - CLion 2019.3.3
  - Visual Studio 2017
- **开发语言**: C语言

# 实现效果

- 计算表达式的值

```
master@cherry-calculator: $ math 1+2+3
The result is 6
```

- 对表达式进行美化

```
master@cherry-calculator: $ math -beauty (   (1  + 2-3)*4 +  (2-4)) /2
After beautify:  (  ( 1 + 2 - 3 )  * 4 +  ( 2 - 4 )  )  / 2
The result is -1
```

- 将中缀表达式转换为后缀表达式

```
master@cherry-calculator: $ math -rev (1+2)*(3+4)-5+6*7/2
The reverse polish notation is: 0 1 2 + 3 4 + * 5 - 6 7 * 2 / +
The result is 37
```

- 表达式中含有非法字符

```
master@cherry-calculator: $ math 1+2sd - 4
The operator we support: [+-*/()], you have input s.
Please check the expression and try again.
```

- 表达式括号不匹配

```
master@cherry-calculator: $ math (1+2)(3+4))
Buckets in the exprssion you input do not match.
Please check the expression and try again.
```

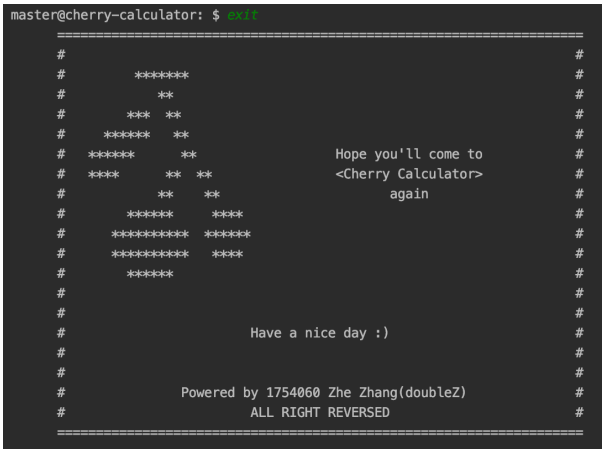
- 帮助信息

```
master@cherry-calculator: $
master@cherry-calculator: $ help
     ============================================================
     #         *******                      Welcome to           #
     #           **                       Cherry Calculator      #
     #        ***  **                                           #
     #     ******   **                    <COMMAND --- math>     #
     #   ******      **          Calculate the value of the expression  #
     #   ****      **  **        and you can do more by adding options. #
     #           **    **          The operator we support: [+-*/()]    #
     #        ******     ****                                    #
     #      **********  ******   [EXAMPLE]                      #
     #      **********   ****       math -beauty 1 +  2   -3*4  #
     #        ******                 math -rev 3 * ( 7 - 4 ) + 8 / 4    #
     #                               math 1+2-(3*4)/5           #
     #         [OPTION LIST]                                     #
     #            -beauty [exp] -> beautify the expression       #
     #            -rev [exp] -> output the reverse polish notation     #
     #            no option -> just calculate the value          #
     #                                                           #
     #              Powered by 1754060 Zhe Zhang(doubleZ)        #
     #                      ALL RIGHT REVERSED                   #
     ============================================================
master@cherry-calculator: $
```

- 未找到该指令

```
master@cherry-calculator: $ sdf
     ============================================================
     #                                                           #
     #         *******                    Welcome to             #
     #           **                     <Cherry Calculator>      #
     #        ***  **                                           #
     #     ******   **                                          #
     #   ******      **                                        #
     #   ****      **  **                    Sorry              #
     #           **    **           Your command is not found   #
     #        ******     ****                                   #
     #      **********  ******                                  #
     #      **********   ****                                   #
     #        ******                                           #
     #                                                         #
     #                                                         #
     #                Input [help] for more information.       #
     #                                                         #
     #                                                         #
     #              Powered by 1754060 Zhe Zhang(doubleZ)      #
     #                      ALL RIGHT REVERSED                 #
     ============================================================
```

- 退出Cherry Calculator




# 核心实现

## 对表达式进行美化

由于用户输入的表达式中可能出现任意数量的空格，因此在获取下一个操作项上要花更多精力进行判断。因此首先将用户输入的字符串进行美化

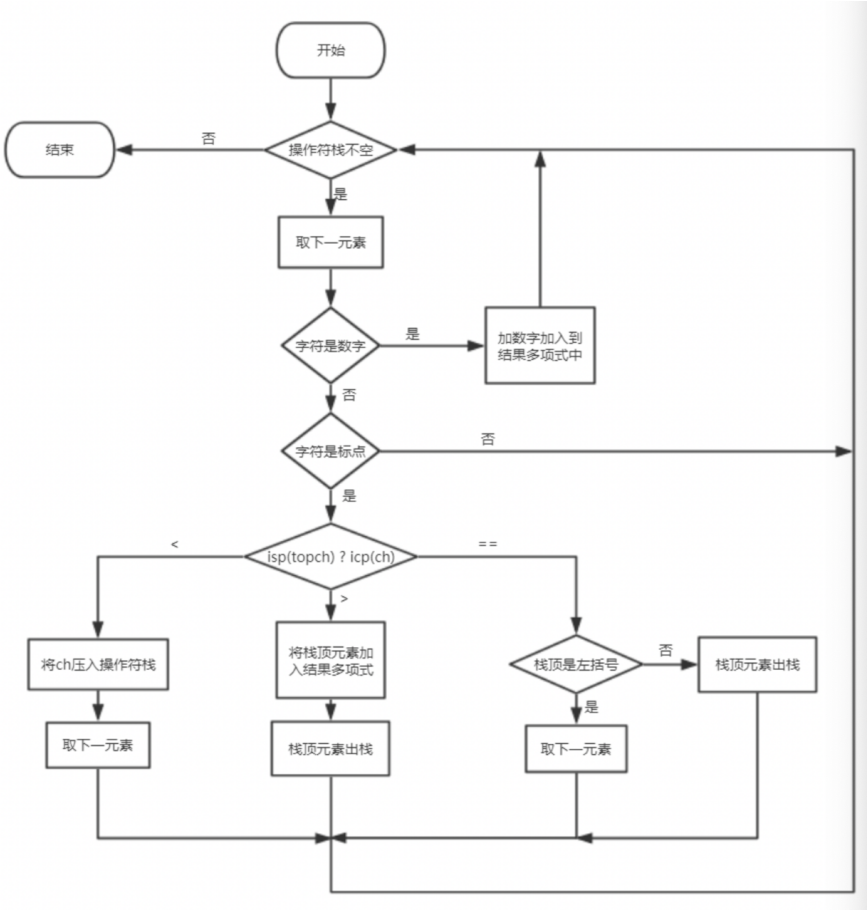
```
/**
 * Beaurify for the origin expression
 * @param origin_exp: origin input expression
 * @param exp: expression after beautify
 */
void beautify(char* origin_exp, char* exp){
    int pos = 0;
    for (int i = 0; i < strlen(origin_exp); ++i) {
        if (isOperator(origin_exp[i])) {
            exp[pos] = ' ';
            ++pos;
            exp[pos] = origin_exp[i];
            ++pos;
            exp[pos] = ' ';
            ++pos;
        }
        else if (isDigit(origin_exp[i])) {
            exp[pos] = origin_exp[i];
            ++pos;
        }
    }
}
```

## 优先级

左括号的栈外优先数最高，它一来到立即进栈，但当它进入栈中后，其栈内优先数变得极低，以便括号内的其他操作符进栈。其他操作符进入栈中后优先数都升1，这样可体现在中缀表达式中相同优先级的操作符从左向右计算的要求，让位于栈顶的操作符先退栈听输出。操作符优先数相等的情况只出现在括号配对或栈顶的'#'号与输入流最后的'#'号配对时。前者将见徐退出位于栈顶的操作符，直到遇到左括号为止。然后将左括号退栈以对消括号，后者将结束算法。

```
/*============ 优先级 ============*/
/**
 * 栈内优先级
 */
int isp(char ch)
{
    switch (ch)
    {
        case '#':return 0;
        case '(':return 1;
        case '*':case '/':return 5;
        case '+':case '-':return 3;
        case ')':return 6;
    }
}

/**
 * 栈外优先级
 */
int icp(char ch)
{
    switch (ch)
    {
        case '#':return 0;
        case '(':return 6;
        case '*':case '/':return 4;
        case '+':case '-':return 2;
        case ')':return 1;
    }
}
```

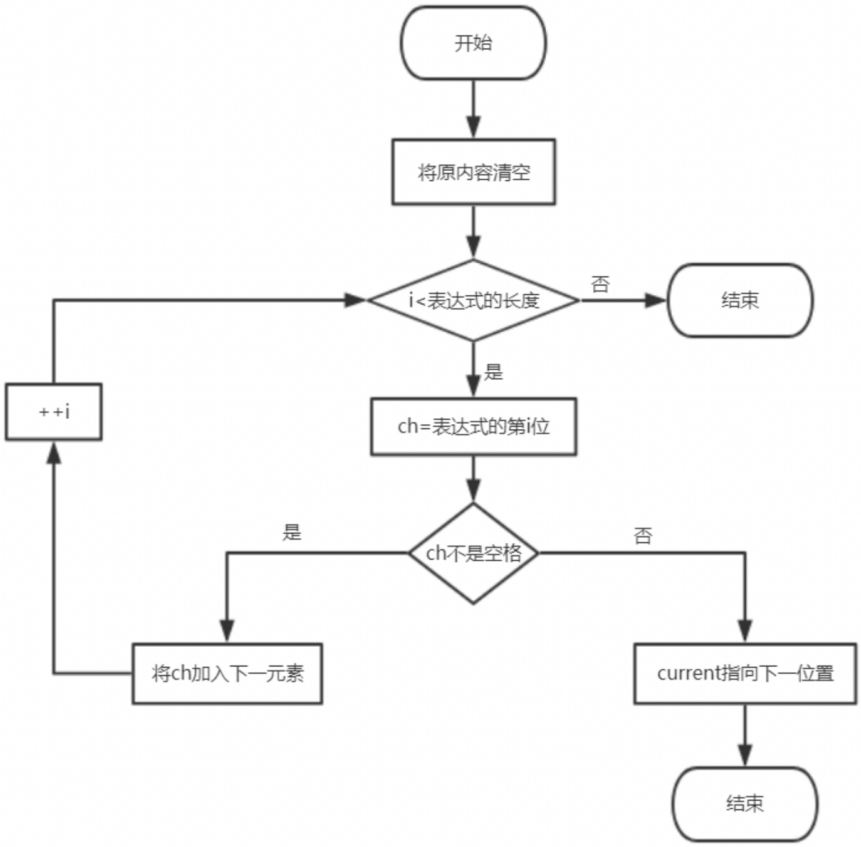
## 将中缀表达式转换为后缀表达式




```
num_stack_clear();  // 初始化操作数栈
op_stack_clear();   // 初始化操作符栈
_current = 0;

struct Data result[100];
int index = 0;

addTail(exp); // 在表达式尾部添加结束标识符

op_stack_push('#');
struct Data elem = NextContent(exp);
while (!isempty_op_stack()) {
  char ch = elem.data;

  if (elem.flag == 1) {         //如果是操作数, 直接读入下一个内容
    result[index] = elem;
    index++;
    elem = NextContent(exp);
  }
  else if (elem.flag == 0) {     //如果是操作符,判断ch的优先级icp和当前栈顶操作符的优先级isp
    char topch = op_stack_top();
    if (isp(topch) < icp(ch)) {         //当前操作符优先级大,将ch压栈,读入下一个内容
      op_stack_push(ch);
      elem = NextContent(exp);
    }
```

```
    else if (isp(topch) > icp(ch)) {     //当前优先级小,推展并输出到结果中
      struct Data buf;
      buf.data = op_stack_pop();
      buf.flag = 0;
      result[index] = buf;
      index++;
    }
    else {
      if (op_stack_top() == '(') {     //如果退出的是左括号则读入下一个内容
        elem = NextContent(exp);
      }
      op_stack_pop();
    }
  }
}
```

## 获取下一内容




## 关于作者

| Item | Info |
| --- | --- |
| **Name** | 张喆 |
| **ID** | 1754060 |
| **Academy** | 软件学院 |
| **Course Name** | 2020人工智能微专业 - 数据结构 |
| **Email** | dbzdbz@tongji.edu.cn |

# 项目结构

```
.
├── CMakeLists.txt          // CLion cmake文件
├── Console-Calculator.exe  // 可执行程序
├── Console-Calculator.sln  // Visual Studio2017 工程
├── README.md
├── doc
│   └── 控制台计算器说明文档.pdf
└── src
    ├── header
    │   ├── calculator.h
    │   ├── calculator_stack.h
    │   ├── interface.h
    │   └── util.h
    └── main.c
```