广州大学学生实验报告

2018年5月26日

**开课学院及实验室：**计算机科学与工程实验室电子楼412A室

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | 计算机 | **专业班级** | 软件171 | **姓名** | 谢金宏 | | **学号** | 1706300001 |
| **实验课程名称** | | 面向对象程序设计实验 | | | | **成绩** | |  |
| **实验项目名称** | | 综合应用程序一 | | | | **指导老师** | | 陶文正 |

# 实验目的及要求

本实验是对前面所学知识的总结，通过一个比较完整的程序的设计，将学过的知识连贯起来，掌握开发一个实际应用程序的步骤，同时学会使用开发工具设计界面友好的应用程序。并通过本实验，掌握如何运用面向对技术对具体的应用系统进行分析和设计。

# 实验内容

制作一个简单的计算器

# 程序代码

程序由三大模块组成：Scanner、Calculator以及Main。Main模块负责用户界面，Scanner负责处理用户输入输出，Calculator执行简单计算器的算法逻辑。代码如下：

// main.cpp

#include "scanner.h"

#include "calculator.h"

**using** **namespace** std**;**

void menu**();**

void help**();**

void credits**();**

int main**()**

**{**

menu**();**

string expression**;**

**while** **(**input**(**expression**))**

**{**

**if** **(**expression **==** "exit"**)** **{**

exit**(**0**);**

**}**

**if** **(**expression **==** "help"**)** **{**

help**();** **continue;**

**}**

**if** **(**expression **==** "credits"**)** **{**

credits**();** **continue;**

**}**

**if** **(**resolve**(**expression**))** **{**

cout **<<** calculate**()** **<<** endl**;**

**}**

**else** cout **<<** "Unable to resolve your input..." **<<** endl**;**

**}**

**}**

void menu**()**

**{**

cout **<<** "Easy Calculator v0.1.0" **<<** endl**;**

cout **<<** R"(Type "help" or "credits" for more information.)" **<<** endl**;**

**}**

void help**()**

**{**

cout **<<** "This calculator can only perform +-\*/ ..." **<<** endl**;**

cout **<<** "It doesn't support operators that accept one argument..." **<<** endl**;**

cout **<<** endl**;**

cout **<<** "Availbe commands:" **<<** endl**;**

cout **<<** "exit - Terminating this calculator" **<<** endl**;**

**}**

void credits**()**

**{**

cout **<<** "Easy Calculator v0.1.0" **<<** endl**;**

cout **<<** "Code by Jinhong Xie(1706300001)." **<<** endl**;**

cout **<<** "Please contact 419246300@qq.com if you have any problem" **<<** endl**;**

**}**

// calculator.h

#include <string>

**using** **namespace** std**;**

bool resolve**(**const string **&**expression**);**

long double calculate**();**

// calculator.cpp

#include <cctype>

#include <iostream>

#include <algorithm>

#include <vector>

#include <map>

#include <stack>

#include "calculator.h"

**using** **namespace** std**;**

struct symbol

**{**

bool isvalue**;**

union **{**

long double value**;** // 操作数

char optype**;** // 操作符

**};**

**};**

void show**(**const symbol **&**syb**)**

**{**

**if** **(**syb**.**isvalue**)** cout **<<** syb**.**value**;**

**else** cout **<<** syb**.**optype**;**

**}**

vector**<**char**>** opavailable **=** **{**

'+'**,** '-'**,** '\*'**,** '/'**,** '('**,** ')'

**};** // 定义运算符

map**<**char**,** int**>** oprank **=** **{**

**{**'+'**,** 2**},** **{**'-'**,** 2**},** **{**'\*'**,** 4**},** **{**'/'**,** 4**},**

**{**'('**,** **-**1**},** **{**')'**,** **-**1**}**

**};** // 定义运算符优先级别

vector**<**symbol**>** symbols**,** prefix\_exp**;**

// 将字符转化为数字符号

auto tonumber **=** **[](**const string **&**expression**,** unsigned **&**idx**)** **{**

long double val **=** 0**;**

long double cur **=** 0.1**;** bool fraction **=** **false;**

**if** **(**expression**[**idx**]** **==** '.'**)** **{**

fraction **=** **true;**

**}**

**else** val **=** expression**[**idx**]-**'0'**;**

**while** **(**idx**+**1**<**expression**.**size**())** **{**

**if** **(**isdigit**(**expression**[**idx**+**1**]))** **{**

**if** **(!**fraction**)** **{**

val **=** val**\***10 **+** expression**[**idx**+**1**]-**'0'**;**

**}**

**else** **{**

val **+=** cur**\*(**expression**[**idx**+**1**]-**'0'**);**

cur **/=** 10**;**

**}**

**++**idx**;**

**}**

**else** **if** **(**expression**[**idx**+**1**]** **==** '.'**)** **{**

**if** **(**fraction**)** **return** **false;**

fraction **=** **true;** **++**idx**;**

**}**

**else** **break;**

**}**

symbols**.**push\_back**({true,** val**});**

**return** **true;**

**};**

// 将字符转化为操作符符号

auto tooptype **=** **[](**const string **&**expression**,** unsigned **&**idx**)** **{**

const decltype**(**opavailable**)** **&**opa **=** opavailable**;**

**if** **(**find**(**opa**.**begin**(),** opa**.**end**(),** expression**[**idx**])** **==** opa**.**end**())**

**return** **false;**

symbols**.**push\_back**(**symbol**{false,** **{.**optype **=** expression**[**idx**]}});**

**return** **true;**

**};**

bool resolve**(**const string **&**expression**)**

**{**

symbols**.**clear**();**

// 解析表达式为符号

**for** **(**unsigned i**=**0**;** i**<**expression**.**size**();** **++**i**)**

**{**

**if** **(**isblank**(**expression**[**i**]))** **continue;**

**if** **(**isdigit**(**expression**[**i**])** **||** expression**[**i**]** **==** '.'**)** **{**

**if** **(**tonumber**(**expression**,** i**))** **continue;**

**else** **return** **false;**

**}**

**else** **{**

**if** **(**tooptype**(**expression**,** i**))** **continue;**

**else** **return** **false;**

**}**

**}**

// 将中缀符号转换为前缀符号

stack**<**symbol**>** opstack**,** midstack**;**

**for** **(**auto it **=** symbols**.**rbegin**();** it **!=** symbols**.**rend**();** **++**it**)**

**{**

// 处理数字

**if** **(**it**->**isvalue**)** **{**

midstack**.**push**(\***it**);** **continue;**

**}**

// 处理括号

**if** **(**it**->**optype **==** ')'**)** **{**

opstack**.**push**(\***it**);** **continue;**

**}**

**if** **(**it**->**optype **==** '('**)** **{**

**while** **(!**opstack**.**empty**()** **&&** opstack**.**top**().**optype **!=** ')'**)** **{**

midstack**.**push**(**opstack**.**top**());**

opstack**.**pop**();**

**}**

**if** **(**opstack**.**empty**())** **return** **false;**

opstack**.**pop**();** **continue;**

**}**

// 处理运算符

**while** **(!**opstack**.**empty**()** **&&** oprank**[**it**->**optype**]** **<** oprank**[**opstack**.**top**().**optype**])**

**{**

midstack**.**push**(**opstack**.**top**());**

opstack**.**pop**();**

**}**

opstack**.**push**(\***it**);**

**}**

**while** **(!**opstack**.**empty**())** **{**

midstack**.**push**(**opstack**.**top**());**

opstack**.**pop**();**

**}**

prefix\_exp**.**clear**();**

**while** **(!**midstack**.**empty**())** **{**

prefix\_exp**.**push\_back**(**midstack**.**top**());**

midstack**.**pop**();**

**}**

**return** **true;**

**}**

long double calculate**()**

**{**

stack**<**long double**>** stk**;**

**for** **(**auto it **=** prefix\_exp**.**rbegin**();** it **!=** prefix\_exp**.**rend**();** **++**it**)**

**{**

**if** **(**it **->** isvalue**)** **{**

stk**.**push**(**it**->**value**);** **continue;**

**}**

long double lhs **=** 0**,** rhs **=** 0**,** rslt **=** 0**;**

**if** **(!**stk**.**empty**())** **{**

lhs **=** stk**.**top**();** stk**.**pop**();**

**}**

**if** **(!**stk**.**empty**())** **{**

rhs **=** stk**.**top**();** stk**.**pop**();**

**}**

**switch** **(**it **->** optype**)** **{**

**case** '+'**:** rslt **=** lhs **+** rhs**;** **break;**

**case** '-'**:** rslt **=** lhs **-** rhs**;** **break;**

**case** '\*'**:** rslt **=** lhs **\*** rhs**;** **break;**

**case** '/'**:** rslt **=** lhs **/** rhs**;** **break;**

**}**

stk**.**push**(**rslt**);**

**}**

**return** stk**.**top**();**

**}**

// scanner.h

#include <iostream>

#include <string>

**using** **namespace** std**;**

void eatline**();**

template **<**typename T**>**

bool input**(**T **&**val**,** const string **&**comment **=** ""**)**

**{**

cout **<<** comment **<<** ">> "**;** cin **>>** val**;**

**return** eatline**();**

**}**

template **<>**

bool input**(**string **&**val**,** const string **&**comment**);**

// scanner.cpp

#include "scanner.h"

**using** **namespace** std**;**

void eatline**()**

**{**

cin**.**clear**();**

**while** **(!**cin**.**eof**()** **&&** cin**.**get**()** **!=** '\n'**)** **continue;**

**}**

template **<>**

bool input**(**string **&**val**,** const string **&**comment**)**

**{**

cout **<<** comment **<<** ">> "**;** getline**(**cin**,** val**);**

**return** **!**cin**.**eof**();**

**}**

# 程序分析

程序启动点设置在Main模块中的main函数。这个函数负责处理用户界面，并根据用户的输入调用各个子模块。main.cpp还包含“Credits”及“Help”函数，负责显示相应的菜单。

Scanner模块包含工具函数input()，负责处理用户输入输出。

简单计算器能够计算用户输入的数学表达式（中缀表达式）。对表达式求值是由Calculator模块负责完成的。求值过程分两步进行，第一步式将用户输入的中缀表达式转换为前缀表达式（resolve），第二步是对前缀表达式求值（calculate）。

将resolve算法归纳如下：

1. 初始化运算符栈S1，中间结果栈S2
2. 从右至左扫描中缀表达式

a) 遇到数字时入栈S2

b) 遇到运算符时比较S1栈顶的优先级，

i) 若S1为空或为')'，运算符入栈 ii) 若运算符优先级比S1栈顶高或相等，运算符入栈 iii) 弹出S1栈顶的运算符至S2，回到b)

c) 遇到括号时，

i) 右括号入栈 ii) 若为左括号，弹出S1中的运算符，并压入S2，直至遇到右括号为止，丢弃左右括号

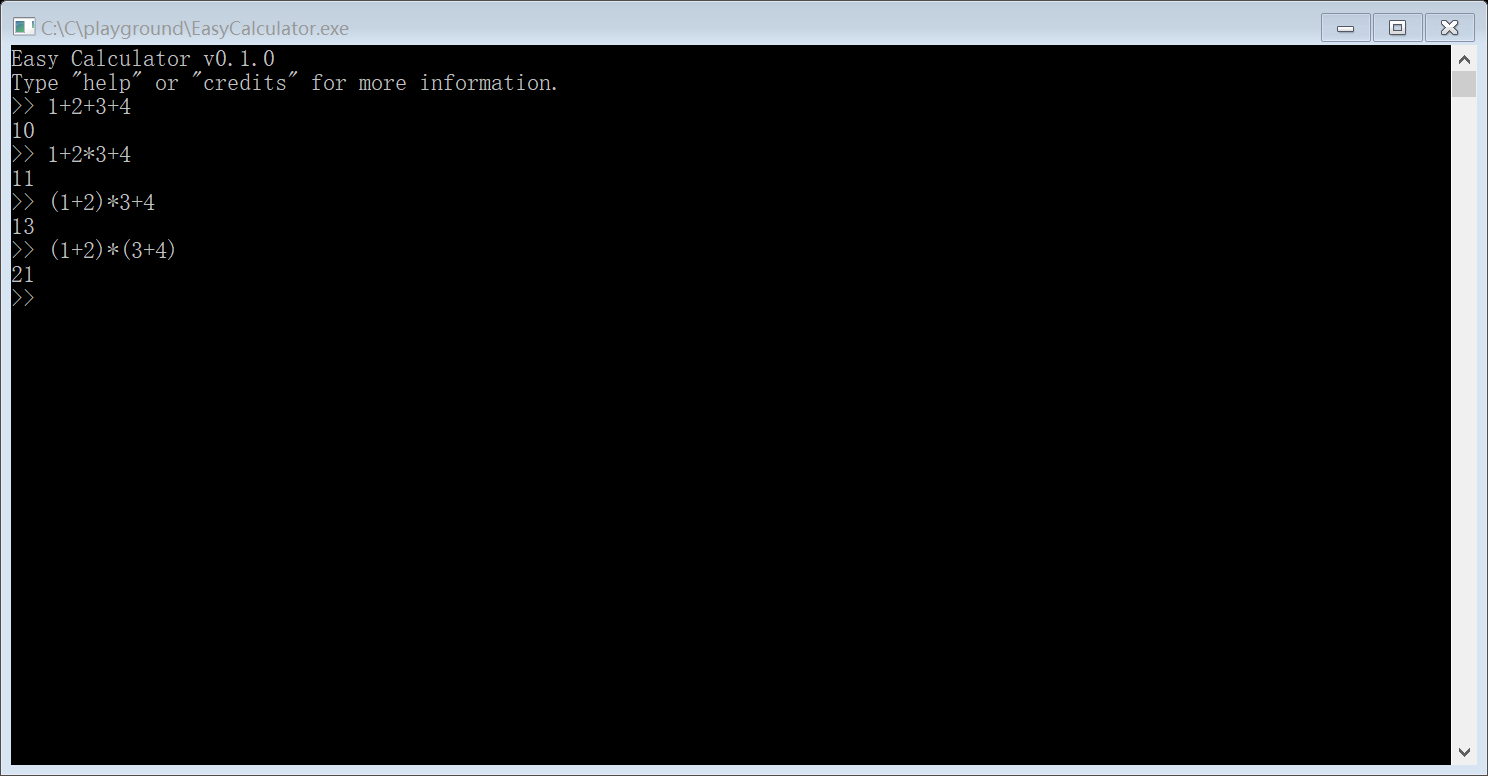
1. 将S1中剩余的运算符弹出并压入S2
2. 弹出S2元素

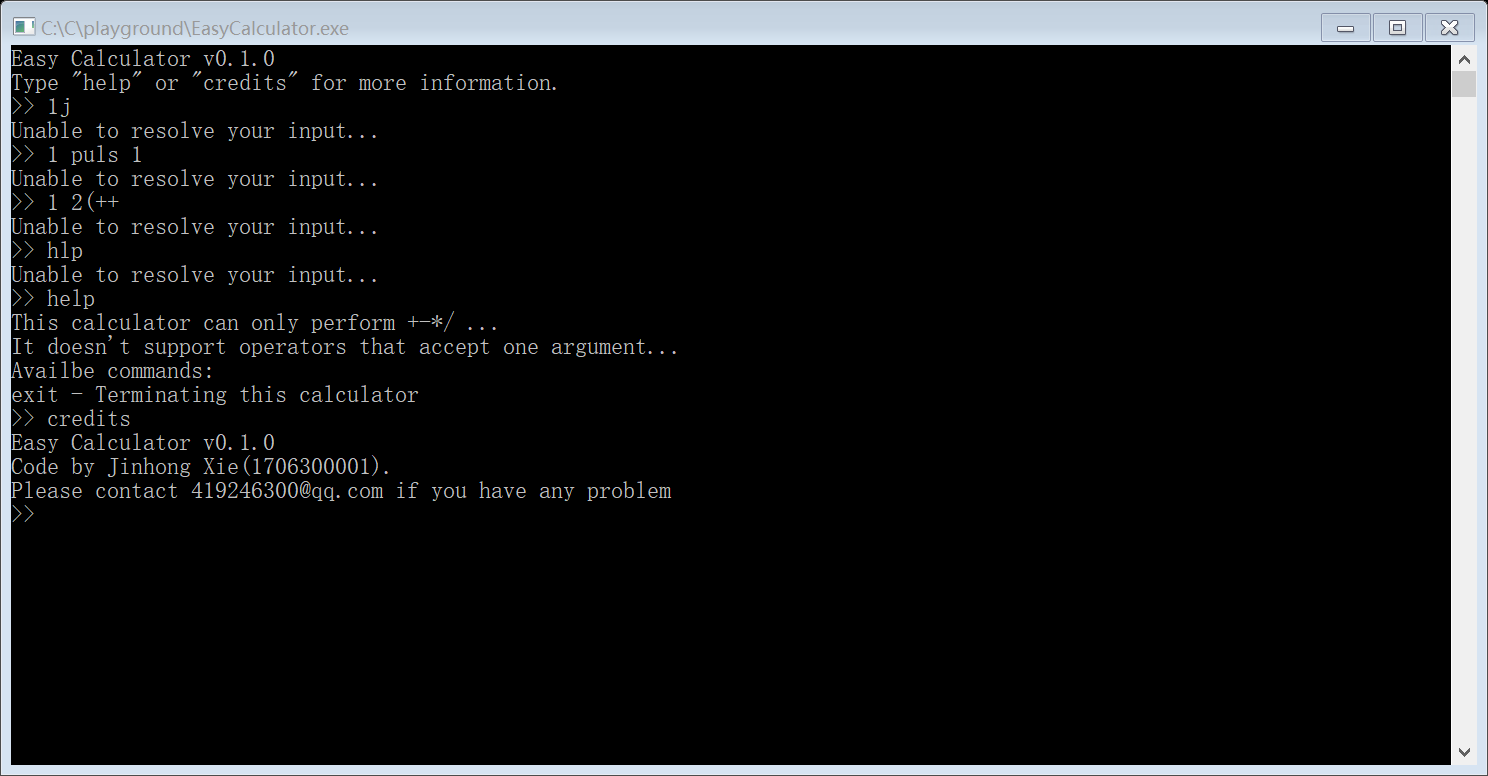
所得即为前缀表达式。

将calculate算法归纳如下：

从右至左扫描前缀表达式，遇到数字入栈，遇到操作符弹出栈顶元素运算(栈顶 op 次顶)，将结果入栈。

# 演示运行





# 总结

程序能够流畅地进行表达式的计算，对用户的错误输入有一定的识别能力，但它仅支持四则运算和括号，并且不支持单目运算符，如符号“-”。