数据结构实验报告

评分

满分——5分

学号：2015111833 姓名：邓金红 专业：计算机科学与技术

知识范畴：栈与队列 完成日期：2017年4月07日

实验题目：后缀表达式求值

实验内容及要求：

从键盘输入后缀表达式（运算符和操作数建议以字符串形式输入，空格作为分隔符），计算并输出后缀表达式的求值结果。

基本要求：实现 +, -, \*, /四个二元运算符；

实现+, -两个一元运算符（即正、负号）；

操作数用整数表示。

提高要求：输出对应的前缀表达式。

每位同学可必须实现基本要求，可选择实现提高要求；程序可不处理表达式语法错误。

实验目的：掌握堆栈在表达式求值中的应用。

数据结构设计简要描述：

使用栈的形式对操作数进行栈入栈出。

算法设计简要描述：

后缀表达式求值的算法实现，扫描输入数据，遇到操作数(整数)则入栈，遇到运算符，这弹出数据进行计算，然后将计算结果入栈。n表示为一元运算符(-)。

输入/输出设计简要描述：

从文件读取输入后缀表达式（以字符串形式输入，空格作为分隔符），计算并输出后缀表达式的求值结果。

编程语言说明：

使用Codeblocks编程。 主要代码采用C语言实现 ；动态存储分配采用C++的new和delete操作符实现；输入与输出采用C++的cin和cout流；程序注释采用C/C++规范。

主要函数说明：

bool isOperator(char c) //判断字符c是否为一个运算符，若是则返回true，否则返回false

int compute(int a, int b, char c) //计算acb的值,比如a=1, b=2, c ='-' 则计算a-b=1-2=-1

int compute\_RPM()//计算后缀表达式整个逻辑函数

程序测试简要报告：

1. **测试实例1*#个位数的后缀表达式***

***程序输入***

2 3 5 \* 3 4 - / + // 2+3\*5/(3-4) = -17

***程序输出***

The computing res is:-17

***结论***

程序输出结果与期望输出结果相符。

1. **测试实例2*#非个位数测试样例***

***程序输入***

12 3 5 \* 3 4 - / + // 12+3\*5/(3-4) = -3

***程序输出***

The computing res is:-3

***结论***

程序输出结果与期望输出结果相符。

1. **测试实例3*#非个位数+一元运算符测试样例***

***程序输入***

12 3 n 5 \* 3 4 - / + // 12+(-3)\*5/(3-4) = 27

***程序输出***

The computing res is:27

***结论***

程序输出结果与期望输出结果相符。

源程序代码：

#include <iostream>

#include <stack>

#include <fstream>

#include <conio.h>

#include <stdlib.h>

#include <stdio.h>

using namespace std;

//2 3 n 5 \* 3 4 - / + 12+(-3)\*5/(3-4) = 17

//判断字符c是否为一个运算符，若是则返回true，否则返回false

bool isOperator(char c)

{

switch(c)

{

case '+':

case '-':

case '\*':

case '/':

return true;

default:

return false;

}

}

/\*\*

参数a, b为操作数， c为操作符，

\*/

int compute(int a, int b, char c) //计算acb的值,比如a=1, b=2, c ='-' 则计算a-b=1-2=-1

{

switch(c)

{

case '+':

return a + b;

break;

case '-':

return a - b;

break;

case '\*':

return a \* b;

break;

case '/':

return a / b;

break;

}

}

int compute\_RPM()//计算后缀表达式主函数

{

FILE \*fp = fopen("input.txt", "r");

stack <int> s;

int a, b;

int result = 0;

char c;

// cin >> noskipws >> c;

c = fgetc(fp);

while(c !=EOF)

{ if(isOperator(c)) //如果是运算符

{

b = s.top(); //弹出栈顶的两个元素，并计算出结果

s.pop();

a = s.top();

s.pop();

if(c == '/' && b == 0) //非法除0操作

{

cout << "Error!Divided bu zero!";

exit(0);

}

result = compute(a, b, c);

s.push(result); //计算结果入栈

}

else if(c == 'n') //一元运算符为符号

{

a = s.top(); //取栈顶元素

s.pop(); //弹出

s.push(-a); //入栈

}else if(isdigit(c))//c为数字操作符

{

result = c - '0';

c = fgetc(fp);

while(isdigit(c))

{

result = result \* 10 + (c - '0');

c = fgetc(fp);

}

s.push(result);

continue;

}

c = fgetc(fp);

}

return s.top();

}

int main()

{

cout << "The computing res is:";

cout << compute\_RPM() << endl;

return 0;

}