山东大学 软件 学院

数据结构 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202000300125 | 姓名：贾星宇 | | 班级：2020级5班 |
| 实验题目：堆栈的应用 | | | |
| 实验学时：4 | | 实验日期：2021年11月9日星期二 | |
| 实验目的：  掌握堆栈的使用。 | | | |
| 硬件环境：  lntel(R) Core(TM) i5-10210U CPU @ 1.60G Hz 2 .1 1 G Hz | | | |
| 软件环境：  CLion 2020.3.3 x64 | | | |
| 实验步骤与内容：  1、 输入一个数学表达式（假定表达式输入格式合法），计算表达式结果并输出。  2、 数学表达式由单个数字和运算符“+”、“-”、“\*”、“/”、“(、) ”构成，例如 2 + 3 \* ( 4 + 5 ) - 6 / 4。  3、 变量、输出采用整数，只舍不入。 | | | |
| 结论分析与体会：  分析堆栈实现计算表达式的算法：  首先，读取对应数据。  需要两个方法：第一个方法判断输入符号的优先级，第二个方法根据符号以及两个数算出结果。  建立两个stack：一个存放数字，一个存放符号，符号区默认一个字符n（null）  计算的优先级为先乘除后加减，有括号则先计算括号内的数字，因此：  利用for循环不断读取字符串中的每一个字符  如果读到数字，则加入到数字的栈区  如果读到符号：  比较该符号与栈内符号优先级：  如果该符号优先级高，则先把该符号放进去，等待下一个数字进入后利用该符号进行两个数字之间的计算  如果该符号优先级低于上一个符号的优先级，则需要先计算上一个符号所需要的两个数字之间的运算，再把该符号插入  即：先把前两个数字弹出，再弹出上一个优先级高的符号，然后进行计算，再弹入下一个符号。  如果遇到左括号，需要设置在栈外左括号优先级最高，先把左括号必须弹入，才能实现有括号先算括号里的。但是弹入左括号后，其优先级不能保持最高，因为这样的话下一个符号，也就是括号内的下一个符号它一定是低优先级，所以一定会先把左括号当\*/一样的弹出然后再弹出两个数运算，所以应该让左括号的优先级变为最低，再不断计算括号内的数字，  直到遇到右括号时，利用循环把括号内的数字按从右向左的方向算出，弹入数字栈内，遇到左括号为止，删掉左括号  例如：（1+2\*3）  左括号优先级大于n，弹入左括号  1，弹入数字栈  +，优先级大于栈内的左括号， 弹入  2，弹入数字  \*，优先级大于+，弹入  3，弹入数字栈  遇到），开始循环  弹出\*，弹出3，2，进行运算  把结果6弹入数字栈，内部是1，6  弹出+，弹出1，6，运算，把7弹入数字栈。  例如：1+2\*3-4  1，弹入数字  +，优先级高于n，弹入  2，弹入数字  \*，优先级高于+，弹入  3，弹入数字  -，优先级低于\*，先运算  弹出2，3，\*，运算结果6弹入数字栈，为1，6  比较-和+，优先级相同，继续计算  弹出1，6，+，计算7，弹入数字栈，为7  比较-和n，优先级-大，弹入-  4，弹入数字  空，利用收尾循环，当符号位不是n时，弹出7，4，-，运算3，弹入数字  符号是n，结束  例如：1\*2\*3：  1，弹入  \*，大于n，弹入  2，弹入  \*，等于\*，计算1\*2  然后和n比较，大，弹入  3，弹入  收尾循环，弹出2，3，\*，计算  符号位为n，结束  结果为6 | | | |
|  | | | |

**本次实验主要代码：**

#include <iostream>

#include <stack>

using namespace std;

int level(int state,char a)

{

int result;

switch (a){

case '+':

case '-':

result = 1;

break;

case '\*':

case '/':

result = 2;

break;

case '(':

if (state == 0) result = 3;

else result = 0;

break;

case 'n':

result = 0;

break;

default:

break;

}

return result;

}

int cal(char option, int num1, int num2)

{

int result;

switch (option){

case '+':

result = num1 + num2;

break;

case '-':

result = num1 - num2;

break;

case '\*':

result = num1\*num2;

break;

case '/':

result = num1/num2;

break;

default:

break;

}

return result;

}

int main() {

cout << "Input"<<endl;

string enter;

cin >> enter;

stack<int> numStack;

stack<char> signStack;

signStack.push('n');

for (int i = 0; i < enter.length(); ++i) {

if (isdigit(enter[i])==true){

int num = enter[i]-48;

numStack.push(num);

} else if (enter[i]=='+'||enter[i]=='-'||enter[i]=='\*'||enter[i]=='/'||enter[i]=='('){

if (level(0,enter[i])> level(1,signStack.top())){

signStack.push(enter[i]);

} else{

while ((level(0,enter[i])<= level(1,signStack.top()))&&signStack.top()!='n'){

int num1 = numStack.top();

numStack.pop();

int num2 = numStack.top();

numStack.pop();

int result = cal(signStack.top(),num2,num1);

signStack.pop();

numStack.push(result);

}

signStack.push(enter[i]);

}

} else if (enter[i]==')'){

while (signStack.top()!='('){

int num1 = numStack.top();

numStack.pop();

int num2 = numStack.top();

numStack.pop();

int result = cal(signStack.top(),num2,num1);

signStack.pop();

numStack.push(result);

}

signStack.pop();

}

}

while (signStack.top()!='n'){

int num1 = numStack.top();

numStack.pop();

int num2 = numStack.top();

numStack.pop();

int result = cal(signStack.top(),num2,num1);

signStack.pop();

numStack.push(result);

}

cout << "Output"<<endl;

cout << numStack.top()<<endl;

cout << "End0" <<endl;

return 0;

}