表达式转换

1651573 刘客

功能分析

- 本项目要求设计程序将中缀表达式转换成后缀表达式
- **输入说明**：输入在一行中给出以空格分隔不同对象的中缀表达式，可包含+, -, *, /以及左右括号，表达式不超过20个字符（不包括空格）。
- **输出说明**：在一行中输出转换后的后缀表达式，要求不同对象（运算数，运算符号）之间以空格分隔，但是结尾不得有多余空格。

编译说明

- 在windows平台下的.exe文件
- 在Linux平台下的out文件

一 . 设计

1. 数据结构设计

> 表达式的分析需要定义各个算符的优先级,所以最先遇到的算符不一定要输出,因此应该使用栈的存储结构来进行这一操作,当栈内算符大于栈外算符优先级的时候,才能输出。
> 故本课程设计采用的主要数据结构是**栈**。

2. 输入输出设计

> 因为此题目规定输入时,各个表达式之间的元素以空格来隔开,所以不能简单地通过cin来进行读取
> 因此本课程设计采用了使用**string**来进行存储,并对string进行处理的方式来解决这一问题。
> 此外输出也特地针对最后一项做了特殊处理,来保证输出时没有空格。

二 . 具体实现细节

1. 处理输入

> 使用getchar读取与string存储相结合的方式
> 当getchar获取的字符不为空格时,进行循环持续读取
> 该方式的好处是能够比较容易的处理 1234,+3 等情况 当这种多个字符连在一起时,使用一般的存储会非常困难

> 最后到获取的字符为'\n'时,则退出循环,读取结束

```
void processCin(string* stringV,int& count) {
    count = 0;
    while (1) {
        char c;
        string tempStr = "";
        bool isEndl = false;//标志是否输入结束
        while ((c = getchar()) != ' ') {
            if (c == '\n') {
                isEndl = true;
                break;
            }
            else {
                tempStr = tempStr + c;
            }
        }
        if (tempStr.size() > 1 && tempStr[0] == '+') {
            tempStr = tempStr.substr(1, tempStr.size());
        }
        stringV[count++] = tempStr;
        if (isEndl) {
            break;
        }
    }
}
```

2. 算符优先级定义

> 因为只有6种算符,故只定义了7个算符(还另外包括'#',来代表一个读取串的开始)的优先级
> 当两个相同级别的元素,分别一个在栈内一个在栈外时,栈内的优先级应该更大一点,因为应该栈内的先输出,所以在定义优先级时,让相同级别的算符在栈内的优先级比在栈外时大1。

> 此处,栈内'(' 的优先级 与 栈外')'的优先级相等,是为了二者相遇时,通过比较可以直接互相消除。

位置	算符	算符	算符	算符	算符	算符
	#	(	+,-	)	*,/	else
In(栈内)	0	1	3	undefined	5	-1
Out(栈外)	undefined	6	2	1	4	-1

> 下面两张图,则是在函数体内定义的栈内和栈外算符的优先级。

3. 栈内栈外字符比较

> 当栈外>栈内,返回1
> 当栈外==栈内 返回0
> 当栈外<栈内 返回-1

```
 //@@returnValue  Type==>int
 // 1 ==> OUT > IN
```

```
// 0 ==> OUT == IN
//-1 ==> OUT < IN
int outCompareIn(string outStr,stack<string>& strStack) {
    //针对无字符输入情况
    if (outStr.compare("") == 0) {
        return -1;
    }
    string inStr = strStack.top();
    int inNum = inStackOp(inStr[0]);
    int outNum = outStackOp(outStr[0]);
    if (outNum > inNum) {
        return 1;
    }
    else if (outNum == inNum) {
        return 0;
    }
    else {
        return -1;
    }
}
```

4. 处理逻辑

> 将表达式读入

> 向栈内push一个文法起始符,保证栈目前不为空

```
strStack.push("#");
```

> 处理逻辑如图

> 针对图中逻辑的一点解释
> 1) 如何判断是否是数字字符串
> 通过string类型的首字符是否为数字字符(isdigit函数)或者string的长度是否大于1(对应情况:110,12或者-2这种字符输入)来判断。
> 2) 栈外算符优先级与栈内算符优先级相等
> 这种情况在自己定义的情况下,只可能在栈内“(”遇到栈外“)”时发生,这时将顶部元素出栈,并直接进行下一次循环,从而处理好括号的情况。 3) 当tempStr为“”的情况
> 在定义优先级的函数里,这种的优先级被定义为-1,即永远不会入栈,所以用“”来处理当读取字符串数组已经完毕但栈内仍有元素的情况。

三 . 程序实现展示

1. 样例1
2. 样例2
3. 样例3
4. 样例4
5. 样例5