《数据结构》课程实践报告

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 院、系 | 计算机学院 | | 年级专业 | 21计科 | 姓名 | 彭光 | 学号 | 2127405052 |
| 实验布置日期 | | 2022-09-14 | | 提交  日期 | 2022-10-22 | | 成绩 |  |

课程实践实验：栈的运算（选做）

## 一、问题描述及要求

设字符串source，依次进入一个初始为空的栈，在入栈的过程中可以出栈，得到一个出栈字符串dest。设计程序，给出所有可能的入栈出栈的过程。假设入栈用i表示，出栈用o表示。

如：

source为：balama

dest为：balama

则输出为：

i o i i i o o i i o o o

i o i i i o o o i o i o

i o i o i o i i i o o o

i o i o i o i o i o i o

source为：value

dest为：lvaeu

则输出为：

输入错误

## 概要设计

对实验内容的理解：

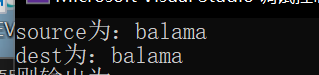
本实验目的是为了考察栈的进栈与出栈的时候元素的顺序问题，首先应该解决掉io的顺序问题，再根据这顺序依次进行进栈与出栈。最后将得到的字符串与dest进行比对，如果有，就输出io的顺序，如果没有，就输出”输入错误”。

系统的功能列表：

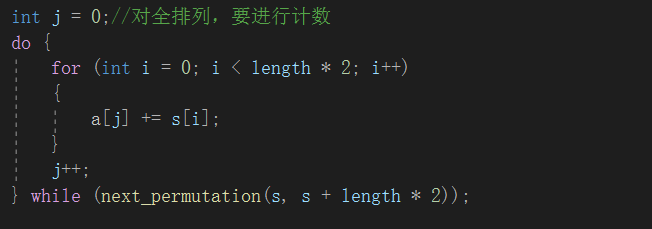
输出对应的入栈与出栈的过程。

程序运行的界面设计：

首先出现屏幕提示，请用户输入source字符串和dest字符串。最后运行出结果



总体设计思路：

首先要将字符串source给一个一个地进栈，再出栈，需要length个push和length个pop（length代表的是source中的char类型数据个数），那么我先建立了一个static的string数组a用来存放可能的进栈出栈过程，然后我用了 

这样的结构进行全排列，利用j来计数，看看有多少种全排列方式。但是数组a里有一些是不可能存在的进栈与出栈过程，所以我编写了一个Istrue函数来判断是否合法。在这个Istrue函数里，我用了一个if语句（if (s[0] == 'o' || s[s.size() - 1] == 'i')//这个判断必须要第一次进栈（因为一开始栈是空的），最后一次出栈

return false;）

和一个while循环(for (int i = 0; i < s.size() - 1; i++)

{

if (s[i] == 'i')//计数i的数目

count\_i++;

else if (s[i] == 'o')//计数o的数目

count\_o++;

if (count\_i < count\_o)//如果出现进栈比出栈少，就是错的，比如“ioo"就是错的，导致下溢。

{

return false;

}

})来完成代码。通过Istrue函数来将那些不符合进栈出栈过程的给删掉。剩下的都是可能的进栈与出栈过程。然后我利用这个情况来进行栈的push和pop，

代码如下：

for (int i = 0; i < k; i++)

{

stack<char>x;

int m = 0;//字符串的下标

for (int n = 0; n < a[i].size(); n++)

{

if (a[i][n] == 'i')

{

x.push(s1[m]);//如果push，m就+1

m++;

}

else if (a[i][n] == 'o')

{

b[i] +=x.top();

x.pop();

}

}

}

将出来的字符串放到数组b中，然后与dest进行匹配，如果有就输出可能的进栈与出栈过程，如果没有就输出“输入错误”。

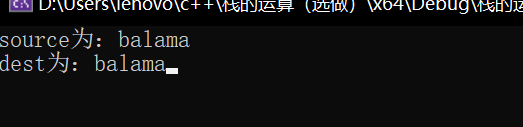
## 详细设计

关键算法设计：对于一个栈将字符串进行不断地进栈出栈要有length个push和length个pop，利用全排列函数，将他们都给存储到数组a中，再利用Istrue函数对不存在的进行删除。最后保留出正确的进栈出栈过程。然后通过这个过程来进行进栈出栈，将最后出来的string给保留出来，最后通过和dest字符串比对，如果有就输出，如果没有就输出“输入错误”。

主函数设计：简单的要求输入字符串source和dest，然后调用process函数进行输出。

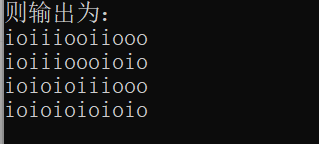
## 实验结果

测试输入：

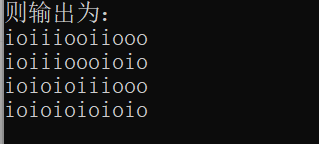


测试目的：测试正常案例是否通过

正确输出：



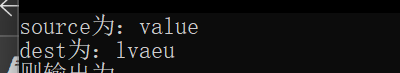
实际输出：



错误原因：无

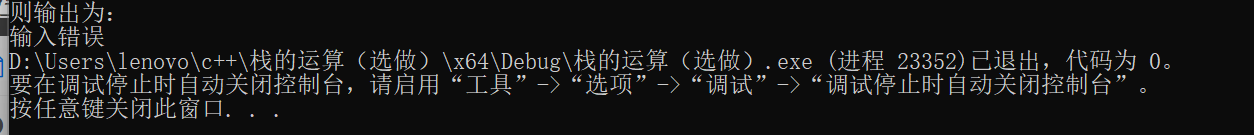
测试结论：通过

测试输入：

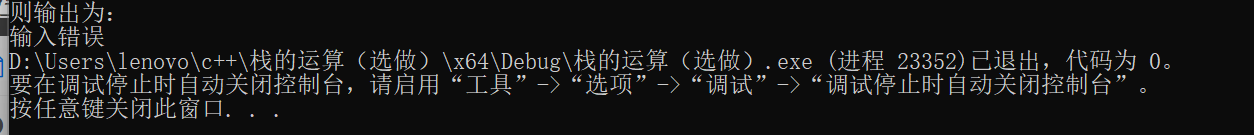


测试目的：测试边界案例是否通过

正确输出：



实际输出：



错误原因：无

测试结论：通过

## 实验分析与探讨

测试结果的分析：运用全排列将所有的列出，最后用函数来将错误的删去。

时间复杂度：O（n\*n!)

空间复杂度：O（n）

## 六、小结

该程序的时间复杂度非常高，我觉得应该要把它降低到O（n\*2)一下。我觉得这个程序写的还是太过简单，主要是偷懒用了个全排列，导致时间复杂度过高。

## 附录：源代码

**1.实验环境：Visual Studio 2022**

（1）源.cpp

#include<iostream>

#include<algorithm>

#include<vector>

#include<stack>

#include<string>

using namespace std;

static string a[1000] = {};//创建一个string数组用来存储类似于“iiiiioooo”这样的stack进出栈的方式（进栈为i，出栈为o）

//Istrue函数是用来判断进出栈是否会出现下溢的情况

bool Istrue(string& s)

{

int count\_i = 0;//用来计数进栈的数目

int count\_o = 0;//用来计数出栈的数目

int length = s.size();

if (s[0] == 'o' || s[s.size() - 1] == 'i')//这个判断必须要第一次进栈（因为一开始栈是空的），最后一次出栈

return false;

for (int i = 0; i < s.size() - 1; i++)

{

if (s[i] == 'i')//计数i的数目

count\_i++;

else if (s[i] == 'o')//计数o的数目

count\_o++;

if (count\_i < count\_o)//如果出现进栈比出栈少，就是错的，比如“ioo"就是错的，导致下溢。

{

return false;

}

}

return true;

}

//process函数是用来比对dest和source字符串通过进出栈是否相同，若相同就输出不同的进出栈方式，不同就输出错误

void process(string& s1, string& s2)//这里面的接口s1是source字符串，s2是dest字符串

{

int length = s1.size();//得出字符串s1的数目

char s[100] = {};//建立一个s的char数组

/\*因为要将一个字符串通过不用的进出栈的方式来匹配新字符串

所以我们要进行length个push和length个pop，

后面要用到全排列，所以我就可以随便选择一种进出栈的方式\*/

for (int i = 0; i < length; i++)

{

s[i] = 'i';

}

for (int i =length; i < length\*2; i++)

{

s[i] = 'o';

}

int j = 0;//对全排列，要进行计数

do {

for (int i = 0; i < length \* 2; i++)

{

a[j] += s[i];

}

j++;

} while (next\_permutation(s, s + length \* 2));

/\*上面是对length个i和o进行全排列，所以将一些错误的也放进了里面。

这里我利用Istrue函数进行判断正误。

\*/

int k = 0;

for (int i = 0; i < j; i++)

{

if (Istrue(a[i]))

{

a[k] = a[i];

k++;

}

}

//已经计算好了正确的push和pop方式；

string b[1000] = {};//这个是用来存储通过进出栈之后返回的字符串，比如说进去的是value，经过进出栈，出来的是vauel，这个数组里存的就是vauel。

for (int i = 0; i < k; i++)

{

stack<char>x;

int m = 0;//字符串的下标

for (int n = 0; n < a[i].size(); n++)

{

if (a[i][n] == 'i')

{

x.push(s1[m]);//如果push，m就+1

m++;

}

else if (a[i][n] == 'o')

{

b[i] +=x.top();

x.pop();

}

}

}

int count = 0;//来对dest和source是否相同来进行计数，如果没有就输出”输入错误“。如果有，就push进c中。

vector<string> c;//用来存储相同字符串时的进出栈方式。

for (int i = 0; i < k; i++)

{

if (b[i] == s2)

{

c.push\_back(a[i]);

count++;

}

}

if (count)//如果count！=0，那么就继续进行输出

{

for (int i = 0; i < c.size(); i++)

{

cout << c[i] << endl;

}

}

else

cout << "输入错误";

}

int main()

{

//输入source和dest

string source, dest;

cout << "source为：";

getline(cin, source);

cout << "dest为：";

getline(cin, dest);

//调用process函数

cout << "则输出为：" << endl;

process(source, dest);

return 0;

}