数据结构与算法栈与队列

陈宇琪

2020 年 3 月 20 日

摘要

主要内容包括 stack 的实现和 stack,queue 的应用。

预备知识：vector，c++ 面向对象，操作符重载，字符串处理。

基本要求：完成 2-5 部分的所有题目，其中第五部分可以不使用面向对象设计思想实现一个简单的 stack。

打 * 的题目为思考题。

自学内容：栈与队列

提交方式：选择题在超星上提交，编程作业在正式上课后选择部分题目打印交上来！同时会有各种考试来检验大家作业的完成情况！

理论 + 应用作业 ddl：2020-03-21，实践作业 ddl：2020-03-18 10:00

目录

1 知识点补充

1.1 栈与队列

栈（stack）

- One of the simplest but most important of all data structures.
- All insertions and deletions of items are made at one end, called the top of the stack.
- With the property called last in, first out, or LIFO

队列（queue）

- One of the simplest but most important of all data structures.
- All additions to the list are made at one end, called the rear or tail of the queue, and all deletions from the list are made at the other end, called the front or head of the queue.
- With the property called first in, first out, or FIFO
- Such as a line of people waiting to purchase tickets.

1.2 Function Templates

对比下面两段代码

Listing 1: **without Fuction Templates**

```
int Abs(int n)
{
        return n<0 ? -n : n ;
}

double Abs(double n)
{
        return n<0 ? -n : n ;
}
```

Listing 2: **using Fuction Templates**

```
template <class Type> Type Abs(Type n)
{
        return n<0 ? -n : n;
}

int main(){
        cout << "Absolute value of -5 is " << Abs(-5);
        cout << endl;
        cout << "Absolute value of -5.6 is " << Abs(-5.6);
        cout << endl;
}
```

2 基础题

1、若让元素 1，2，3，4，5 依次进栈，则出栈次序不可能出现在（）种情况。

A．5，4，3，2，1 B．2，1，5，4，3 C．4，3，1，2，5 D．2，3，5，4，1

2、已知一个栈的入栈序列是 1，2，3，…，n，其输出序列为 p1，p2，p3，…，pn，若 p1=n，则 pi 为（）。

A．i B．n-i C．n-i+1 D．不确定

3、设栈 S 和队列 Q 的初始状态为空，元素 e1、e2、e3、e4、e5 和 e6 依次进入栈 S，一个元素出栈后即进入 Q，若 6 个元素出队的序列是 e2、e4、e3、e6、e5 和 e1，则栈 S 的容量至少应该是（ ）。

A．2 B．3 C．4 D．6

3 简答题

1、用数组实现队列有什么不足的地方？是否可以用 vector 实现一个 $O(1)$ 插入，$O(1)$ 删除的队列？

*2、如何生成一个合法的出栈顺序？（需要用到一些 random 的事情）

（这个问题看上去很抽象，我来给大家解释一下：在 EOJ 里面你们会看到一道题目叫“铁路调度”，为了给大家构造测试数据，我需要生成若干合法的出栈顺序，若干不合法的出栈顺序，用怎么样的一个算法可是使得生成出来的数据在尽可能多样的条件下，部分数据是合法的，部分数据是非法的。说的更明白一点的话：因为我们知道绝大多数的出栈顺序都是非法的，如果直接随机生成的话可能导致绝大多数数据都是非法的，甚至可能直接输出 no 就可以通过（大家可以自己体验一下），如果精心构造每一个样例又很浪费时间，所以我们需要一个算法来生成一个很强的数据。）

备注：有兴趣的同学可以在 EOJ 上挑战 1.5。

4 基础编程题

1、从键盘上输入一个后缀表达式，试编写算法计算表达式的值。规定：逆波兰表达式的长度不超过一行，操作数之间用空格分隔, 操作符只可能有 +、-、*、/四种运算。例如 3 4 + 2 * 。（注意特殊情况）。

备注：参考 EOJ 题库 1.2 10 以内的后缀表达式求解

2、给定一个括号匹配串（定义可以参考“应用 1”题目），输出每个右括号对应的左括号。（字符串从 1 开始编号）。例如输入”(()())”，输出” 2 4 1”（说明样例中在编号 3 的右括号对应编号 2 的左括号；在编号 5 的右括号对应编号 4 的左括号；在编号 6 的右括号对应编号 1 的左括号）

3、完成 EOJ 题库中 1.2-1.3 以及 2.2-2.3 题目。

*4、我们一般使用的表示式并不是前面看到的后缀表达式，而是中缀表达式，例如 (3+4)*2，设计算法将中缀表达式转换成后缀表达式（使用栈的相关知识）。

说明

提交要求：完整代码 + 运行截图。

5 stack 实现

利用 c++ 现有的 vector 库实现 stack。

基本要求：学习 vector 的使用方法，实现除重载运算符之外的所有操作，**可以不使用 template 和 class**。

Listing 3: **stack.cpp**

参考代码

```
#include <vector>
```

```
#include <iostream>
using namespace std;

template<class T>
class stack
{

public:
        stack();
        ~stack();
        T top();
        void pop();
        void push(const T &item);
        int size();
        bool empty();
        stack<T>& operator = (const stack<T>& rhs);
        friend ostream& operator << (ostream& os, stack<T>& s);

private:
        vector <T> entry;
        int count;
};

template<class T>
stack<T>& stack<T>::operator = (const stack<T>& rhs)
{
        //Copy contents of rhs
        //Remain rhs unchanged
}

template<class T>
ostream& operator << (ostream& os, stack<T>& s)
{
        //Output elements of s in the order LIFO
        //Remain s unchanged
}

int main()
{
        stack<int> s;
        s.push(1);
        s.push(2);
        cout<<s.top()<<endl;
        cout<<s<<endl;
        s.pop();
        cout<<s<<endl;
        stack<int> t;
        s.push(3);
        t=s;
        cout<<t<<endl;
```

```
        t.push(4);
        s.push(5);
        cout<<s<<endl;
        cout<<t<<endl;
        s.pop();
        cout<<s<<endl;
        cout<<t<<endl;
        stack<int> u;
        u=t;
        cout<<u.size()<<endl;
        while (!u.empty())
        {
                cout<<u.top()<<' ';
                u.pop();
        }
        return 0;
}

/*
2
2 1
1
3 1
5 3 1
4 3 1
3 1
4 3 1
3
4 3 1
*/
```

大家也可以按照 C++ Reference 实现。

Member types

C++98 C++11

member type	definition	notes
value_type	The first template parameter (T)	Type of the elements
container_type	The second template parameter (Container)	Type of the *underlying container*
size_type	an unsigned integral type	usually the same as size_t

Member functions

(constructor)	Construct stack (public member function)
empty	Test whether container is empty (public member function)
size	Return size (public member function)
top	Access next element (public member function)
push	Insert element (public member function)
emplace C++11	Construct and insert element (public member function)
pop	Remove top element (public member function)
swap C++11	Swap contents (public member function)

Non-member function overloads

relational operators	Relational operators for stack (function)
swap (stack) C++11	Exchange contents of stacks (public member function)

Non-member class specializations

uses_allocator<stack> C++11	Uses allocator for stack (class template)

图 1: C++ Reference for Stack

基本要求（Baseline）

用 C/c++ 自己实现一个 stack。

附加分

class 封装/template 模板/重载运算符/.....

6 stack 应用

提示：在该部分可以使用 stl 的 stack。

题意

给你一个由'('、')' 和小写字母组成的字符串 s。

你需要从字符串中删除最少数目的'(' 或者')' （可以删除任意位置的括号)，使得剩下的「括号字符串」有效。

请返回任意一个合法字符串。

有效「括号字符串」应当符合以下 **任意一条**要求：

- 空字符串或只包含小写字母的字符串
- 可以被写作 AB（A 连接 B）的字符串，其中 A 和 B 都是有效「括号字符串」
- 可以被写作 (A) 的字符串，其中 A 是一个有效的「括号字符串」

输入

s = "lee(t(c)o)de)"

输出

"lee(t(c)o)de"

解释

"lee(t(co)de)" , "lee(t(c)ode)" 也是一个可行答案。

提交链接：https://leetcode-cn.com/problems/minimum-remove-to-make-valid-parentheses/

关于栈和队列各种应用可以参考 LeetCode 上的题目：

https://leetcode-cn.com/problemset/all/?topicSlugs=stack
https://leetcode-cn.com/problemset/all/?topicSlugs=queue
https://leetcode-cn.com/problemset/all/?topicSlugs=breadth-first-search

7 参考答案

7.1 选择题答案

CCB

7.2 简答题答案

1、关键点：用数组实现的 queue 会浪费空间（pop 的实现无法做到真正释放内存空间）；使用 vector 的 erase 函数会使得 pop 的复杂度变成 $O(N)$。

7.3 基础编程题答案

1、算法描述：维护一个 stack<int>，遇到数就放进 stack 中，遇到操作符就从 stack 中拿出两个数进行运算（注意前后顺序）。

2、注意输入的字符串不一定合法！

Listing 4: **ans.cpp**

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
	string t;
	stack<int> s;
	cin>>t;
	vector <int> ans;
	for (int i=0;i<t.length();i++)
	{
		if (t[i]=='(')
			s.push(i);
		else if (t[i]==')')
		{
			if (s.empty()) //note!!
			{
				cout<<"Input Error"<<endl;
				return 0;
			}
			else
			{
				ans.push_back(s.top()+1);
				s.pop();
			}
		}
		else
		{
			cout<<"Input Error"<<endl;
			return 0;
		}
	}
	if (!s.empty())
```

```
        {
                cout<<"Input Error"<<endl;
                return 0;
        }
        for (int i=0;i<ans.size();i++)
                cout<<ans[i]<<' ';
        return 0;
}
```

4、算法描述：可以先将中缀表达式转成后缀表达式，这里要特别注意运算符的优先级问题！转成后缀表达式之后就可以进行求值。当然直接做也可以，就是代码会很难写！

7.4 stack 实现参考答案

参考答案（By 范胤杰）：亮点是使用了 try-catch 做异常处理，但是问题在于 top 函数一般是要返回栈顶元素的，而不是 void 类型。

Listing 5: **ans.cpp**

```
#include<iostream>
#include<sstream>
#include<string>
#include<vector>

using namespace std;

template<class T>
class Stack{
public:
    Stack();
    Stack(const Stack& rhs);
    void pop();
    void push(const T& x);
    void vacant()   const;
    int capacity()  const;
    void top()  const;
    void flush();

private:
    vector<T>elems;
};

template<class T>
inline Stack<T>::Stack()
: elems()
{   }

template<class T>
inline Stack<T>::Stack(const Stack& rhs)
: elems(rhs.elems)
{   }
```

```
template<class T>
void Stack<T>::pop()
{
    try{
        if(elems.empty())
            throw "ERROR";
        else{
            elems.pop_back();
        }
    }
    catch(char const* str){
        cout << str << endl;
    }
}

template<class T>
inline void Stack<T>::push(const T& x)
{
    elems.push_back(x);
}

template<class T>
void Stack<T>::vacant() const
{
    if(elems.empty()){
        cout << "YES" << endl;
    }else{
        cout << "NO" << endl;
    }
}

template<class T>
inline int Stack<T>::capacity() const
{
    return elems.size();
}

template<class T>
void Stack<T>::top() const
{
    try{
        if(elems.empty()){
            throw "ERROR";
        }else{
            cout << elems[elems.size()-1] << endl;
        }
    }
    catch(char const* str){
        cout << str << endl;
```

```
    }
}

template<class T>
inline void Stack<T>::flush()
{
    elems.clear();
}

int main(void)
{
    int Q;
    Stack<int>sequence;
    string order;
    cin.tie(0);
    ios::sync_with_stdio(0);
    cin >> Q;
    cin.get();
    for(int i = 0; i < Q; i++){
        getline(cin,order);
        if(order == "pop"){
            sequence.pop();
        }else if(order == "top"){
            sequence.top();
        }else if(order == "size"){
            cout << sequence.capacity() << endl;
        }else if(order == "clear"){
            sequence.flush();
        }else if(order == "empty"){
            sequence.vacant();
        }else{
            istringstream stream(order);
            int temp;
            stream >> order;
            stream >> temp;
            sequence.push(temp);
        }
    }
}
```

8 作业点评

1、简答题有些同学回答过于简单，甚至就几个关键字。

2、编程题尽量能够考虑到程序的鲁棒性（例如：**非法的输入**）。

3、部分同学没有使用 string 而是使用了数组，并且数组长度较小，似乎也不是很妥，一般可能开到 10^5 数量级可能较妥一点。

4、个别由于编程题出现 bug 导致分数低于 95 分的同学，请认真阅读超星上的作业批语。