中 国 矿 业 大 学

2018 级《数据结构与算法分析》课程作业

学生姓名 王茂凯

学　　号 04181425

中国矿业大学信控学院

1. 铁路进行列车调度时，常把站台设计成栈式结构的站台，如图所示。试问：

(1)设有编号为 1,2,3,4,5,6 的 6 辆列车，顺序开入栈式结构的站台，则可能的出栈序列有多少种？

一辆列车　　1 种

两辆列车 可分为前一辆和后一辆　　2 种

三辆列车 可整体看成一辆,或前后分成两辆　1+2+2　　5 种

```
#include <iostream>
using namespace std;
void solution(const int &n)
{
    int *t = new int[n + 1]();
    t[0] = 1;
    for (int i = 1; i <= n; i++)
    {
        if (!t[i])
        {
            for (int j = 0; j <= i - 1; j++)
                t[i] += t[j] * t[i - 1 - j];
        }
    }
    cout << t[n] << endl;
    delete[] t;
}
int main()
{
    int n = 0;
    cin >> n;
    solution(n);
    return 0;
}
```

```
6
132
```

(2)若进站的 6 辆列车顺序如上述所示，那么是否能够得到 435612，325641，154623 和 135426 的出站序列，如果不能，说明为什么不能；

如果能，说明如何得到（即写出“进栈”或“出栈”的序列）。

435612　不能,刚开始 1234 入栈 ,出栈时 2 一定在 1 前
325641　123 入栈,23 出栈,4 入栈,5 入栈,5 出栈,6 入栈,6 出栈,4 出栈,1 出栈
154623　不能,1 入栈,2 入栈,3 入栈后,出栈时 3 在 2 前
135426　123 入栈,3 出栈,4 入栈,5 入栈,5 出栈,4 出栈,2 出栈,6 入栈,6 出栈,1 出栈

2 写出以下中缀表达式的后缀表达式:

(1) A×B×C

(2) –A+B-C+D

(3) (A+B) ×D+E/(F+A×D)+C

(4) A×B–C/B^2

算法实现

```
A B * C *
A - B + C - D +
A B + D * E F A D * + / + C +
A B * C B 2 ^ / -
```

```
#include <iostream>
#include <stack>
#include <string>
using namespace std;
/*
由中缀表达式求后缀表达式的关键
判断栈顶运算符与将要入栈运算符的优先级:(乘除优先于加减)
将要入栈运算符的优先级若小于等于栈顶运算符的优先级,则输出栈顶运算符
循环判断直到栈空或遇到左括号
将将要入栈的运算符入栈
*/
void solution(string s)
{
    stack<char> strstack;
    int len=s.length();
    for (int i = 0; i < len; ++i)
    {
        char t = s[i];
        if (((int)s[i] < 40 || (int)s[i] > 47) && (int)s[i]!=94)//ACSII
码方便判断
            cout << s[i] << " ";
```

```
        else if (strstack.empty() || t == '(' || t=='^') //栈为空或字符为 (  ^ 直接入栈
            strstack.push(t);
        //关键
        else
        {
            while (!strstack.empty())
            {
                char temp = strstack.top(); //栈顶运算符赋值给 temp
                if (t == '+' || t == '-')
                {
                    if (temp == '(')
                        break;
                    strstack.pop();
                    cout << temp << " ";
                }
                else if (t == '*' || t == '/')
                {
                    if (temp == '*' || temp == '/' || temp == '^')
                    {
                        strstack.pop();
                        cout << temp << " ";
                    }
                    else
                        break;
                }
                else if (t == ')')
                {
                    strstack.pop();
                    if (temp == '(')
                        break;
                    cout << temp << " ";
                }
            }
            if (t != ')')
                strstack.push(t);
        }
    }
    while (!strstack.empty()) //将栈中的剩余运算符输出
    {
        char temp = strstack.top();
        cout << temp << " ";
        strstack.pop();
    }
```

```
    cout << endl;
}
int main()
{
    solution("A*B*C");
    solution("-A+B-C+D");
    solution("(A+B)*D+E/(F+A*D)+C");
    solution("A*B-C/B^2");
    return 0;
}
```

3. 实现数存放循环队列。以 front 和 rear 为队列的队首队尾。请实现相应的入队和出队元素的操作。（请附上可执行的代码）

```
#include <iostream>
using namespace std;
class Queue //循环队列
{
public:
    Queue(int size = 5)
    {
        _pQue = new int[size]();
        _front = 0;
        _rear = 0;
        _size = size;
    }
    ~Queue()
    {
        delete[] _pQue;
        _pQue = nullptr;
    }
    void push(int val) //入队
    {
        if (full())
            resize();
        _pQue[_rear] = val;
        _rear = (_rear + 1) % _size;
    }
    void pop() //出队
    {
```

```
        if (empty())
            return;
        _front = (_front + 1) % _size;
    }
    bool full() { return (_rear + 1) % _size == _front; }
    bool empty() { return _front == _rear; }
    void print()
    {
        if (empty())
            return;
        int f = _front;
        int r = _rear;
        while (f % _size != r)
        {
            cout << _pQue[f] << " ";
            f = (f + 1) % _size;
        }
        cout << endl;
    }

private:
    int *_pQue;
    int _front; //指向队头
    int _rear;  //指向队尾
    int _size;  //空间

    void resize() //二倍扩容
    {
        int *ptmp = new int[_size * 2]();
        int index = 0;
        for (int i = _front; i != _rear; i = (i + 1) % _size)
        {
            ptmp[index++] = _pQue[i];
        }
        delete[] _pQue;
        _pQue = ptmp;
        _front = 0;
        _rear = index;
        _size *= 2;
    }
};
int main()
{
    Queue que;
```

```
    for (int i = 0; i < 15; ++i) //入队 15 个随机数
        que.push(rand() % 10 + 1);
    que.print();  //打印
    for (int i = 0; i < 5; ++i) //前 5 个数出队
        que.pop();
    que.print(); //打印
    return 0;
}
```

```
2 8 5 1 10 5 9 9 3 5 6 6 2 8 2
5 9 9 3 5 6 6 2 8 2
```