**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;

}



**(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:**

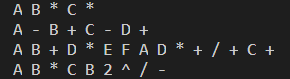
**(1) A×B×C**

**(2) –A+B-C+D**

**(3) (A+B) ×D+E/(F+A×D)+C**

**(4) A×B–C/B^2**

**算法实现**



#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;

}

