Lesson 3 Practice:

221900180 田永铭

程序填空:

1. 设ha和hb分别是两个带表头结点的**非递减有序**单链表的表头指针, 试设计一个算法, 将这两个有序链表合并成一个**非递增有序**的单链表。要求结果链表仍使用原来两个链表的存储空间, 不另外占用其它的存储空间。表中允许有重复的数据。

template <class Type>

void *List* <Type> :: *Merge* ( *List*<Type>& *hb*) {

//将当前链表this与链表*hb*按逆序合并，结果放在当前链表this中。ListNode<Type> \*pa, \*pb, \*q, \*p;pa = first→link; pb = hb.first→link; //检测指针跳过表头结点

first→link = NULL; //结果链表初始化

while ( pa != NULL && (1) pb!=NULL ) { if ( pa→data <= pb→data )

{ q = pa; (2) pa=pa->link ; }

else

{ q = pb; (3) pb = pb->link ; }

(4) q->link = first->link ; first→link = q;

}

p = (5) (pa!=NULL) ? pa : pb ;

while ( (6) p!=NULL ) { q = p; p = p→link;

(7) q->link = first->link ;

(8) first->link = q ;

}

}

2. 写出下列中缀表达式的后缀形式：

(1) A \* B \* C

(2) - A + B - C + D

(3) A\* - B + C

(4) (A + B) \* D + E / (F + A \* D) + C

(5) A **&&** B|| ! (E > F) /\*注：按C++的优先级\*/

(6) !(A **&&** !( (B < C)||(C > D) ) )||(C < E) \

答：（1）A B \* C \*

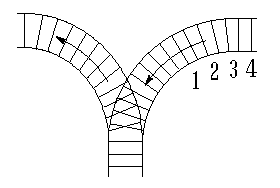
（2）A – B + C – D +

（3）A 0 B - \* C +

（4）A B + D \* E F A D \* + / + C +

（5）A B && E F > ! ||

（6）A B C < C D > || ! && ! C E < ||

3. 铁路进行列车调度时, 常把站台设计成栈式结构的站台，如右图所示。试问：

(1) 设有编号为1,2,3,4,5,6的六辆列车, 顺序开入栈式结构的站台, 则可能的出栈序列有多少种?

(2) 若进站的六辆列车顺序如上所述, 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

答：

1. CatLan(6) = (2\*6)!/((6!) \* (7!)) = 132 种
2. (i)435612：不能，4的先出预示着1在2底下，所以不能出现12序列

(ii)325641：能，方法如下：进1，进2，进3，出3，出2，进4，进5，出5，进6，出6，出4，出1

(iii)154623:不能，先进1，出1，然后就出了5意味着从栈底到栈顶依次是234，不可能出成23

(iv)135426:能，方法如下：进1，出1，进2，进3，出3，进4，进5，出5，出4，出2，进6，出6

4. 设表达式的中缀表示为a \* x - b / x↑2；(计算的优先级是↑>\*/>+-)，试利用栈将它改为后缀表示ax \* bx2↑/ -。若在转换过程中，遇到操作数直接输出并读下一字符，遇到操作符进行入栈并读取下一字符以及出栈并输出操作，请考虑在转换的过程中何时执行入栈，何时执行出栈，写出转换过程中的入栈（用I表示）和出栈（用O表示）的序列。

答案：stack<char> opstack; string ans;

从左往右读：

（1）读到a，直接加入ans. opstack:空 ans = “a”

（2）读到\* ，入栈I opstack:\* ans = “a”

（3）读到x,直接加入ans opstack:\* ans = “a x”

（4）读到-，-号比\*优先级小，先O后I opstack：- ans = “a x \*”

（5）读到b, 直接加入ans opstack:- ans = “a x \* b”

（6）读到/，/比-优先级大，I opstack:- / ans = “a x \* b”

（7）读到x，直接加入ans opstack:- / ans = “a x \* b x”

（8）读到↑，优先级比栈内元素都大，I opstack:-/ ↑ ans = “a x \* b x”

（9）读到2，直接加入ans opstack:-/ ↑ ans = “a x \* b x 2”

（10）都结束，依次pop出栈 ans = “a x \* b x 2 ↑ / -”

栈操作序列为： I O I I I O O O

I：\* - / ↑

O： \* ↑ / -

5. 如果用一个循环数组q[0..m-1]表示队列时，该队列只有一个队列头指针front，不设队列尾指针rear，而改置计数器count用以记录队列中结点的个数。

（1）编写实现队列的三个基本运算：判空、入队、出队。

（2）队列中能容纳元素的最多个数是多少？

**typedef** **struct**

{ TYPE q[m];

**int**front,count;  //front是队首指针，count是队列中元素个数。

} cqnode;           //定义类型标识符。

答案：

(1)

判空:bool empty(cqnode cq)

{

return (cq.count == 0);

}

入队: bool EnQueue(cqnode cq,TYPE x)

{

if(count==m){printf(“队满\n”);return false;}

cq.q[(cq.front + cq.count)%m]=x;

cq.count++ ;

return true;

}

出队: bool OutQueue(cqnode cq,TYPE x)

{

if (cq.empty()){printf(“队空\n”);return false;}

x=cq.q[cq.front];

cq.front=(cq.front+1)%m;

cq.count--;

return true;

}

(2) m个