

栈与队列

代码题 1

题干

- 题目：

栈的结构体：

```
typedef struct
{
    char data[maxsize];
    int top;
} stack;
```

队列的结构体

```
typedef struct
{
    int data[maxsize];
    int front, rear;
} queue;
```

- 1、Q 是一个队列，S 是一个空栈，编写算法使得队列中元素逆置

- 总结：

- 将 Q 中全部元素依次出队，出队后进入 S；
- Q 中元素全部进入 S 后，依次出栈、进入队列 Q；

解答：方法一

```
void Invert(queue* &queue, stack* &stack)
{
    while(queue.front != queue.rear)
    {
        // front表示当前队头，在数组的低位置。出队时、front往前移动，front也要自加1
        int data = queue.data[front++];

        // stack默认为空，因此top默认指向-1，如果要入栈，需要先对top自加，然后将元素放到对应位置
        stack.data[++top] = data;
    }
}
```

```

while(stack.top!=-1)
{
    // 取出栈顶元素
    int data = stack.data[top--];

    // 将栈顶元素入队
    queue.data[rear++] = data;
}
}

```

解答：方法二

```

void Invert(queue &queue, stack &stack)
{
    while(queue.front!=queue.rear)
    {
        push(stack, dequeue(queue));
    }

    while(stack.top!=-1)
    {
        enqueue(queue, pop(stack));
    }
}

```

代码题 2

题干

- 题目：
 - 2、判断单链表的全部 n 个字符是否中心对称
- 总结：
 - 第一步：遍历单链表，确定链表长度，获得链表的中心位置 $n/2$
 - 第二步：将前一半的元素入栈，同时指针停留在链表的中心位置
 - 第三步：将栈内元素出栈的同时、指向中心位置的指针往后移动，逐个比较是否相同，如果不相同则为 `false`；

解答

```

bool IsSymmetry(LNode* list)
{
    // 第一步
    LNode* p = list->next;
    int length = 0;

    while(p!=null)
    {
        length++;
    }
}

```

```

        p=p->next;
    }

    // 第二步
    stack stack;
    Init(stack);
    p = list->next;

    for(int i =1 ; i<=length/2 ;i++)
    {
        push(stack,p->data);
        p = p->next;
    }
    // 循环结束后，当前指针p在第length/2个位置

    // 第三步
    // 如果是奇数，则最中间的元素无需比较，让p往后移动一格
    if(length*2==1)
    {
        p = p->next;
    }
    while(p!=null)
    {
        if(p->data!=pop(s))
        {
            return false;
        }
        p = p->next;
    }
    return true;
}

```

代码题 3

题干

- 题目：
 - 3、两个栈 s1,s2 都采用顺序存储，并共享一个存储区[0,...,maxsize-1]。采用栈顶相向，迎面增长的存储方式，设计 s1,s2 入栈和出栈的操作。
- 总结：
 - 入栈：需要注意判断当前两个栈的栈顶是否相遇
 - 出栈：需要判断两个栈是否到底

解答

```

// i 用来选择哪一个栈。0表示对s1压栈，1表示对s2压栈
void push(stack &stack,int i,int data)

```

```

{
    if(i>1 || i<0)
    {
        return;
    }

    // 当前到达栈顶
    if(stack.top[1]-stack.top[0]==1)
    {
        return;
    }

    switch(i)
    {
        case 0:
            // s1从低地址往高地址增长
            stack.data[++stack.top[0]];
            break;

        case 1:
            // s2从高地址往低地址增长
            stack.data[--stack.top[1]];
            break;
    }
}

int pop(stack &stack,int i)
{
    if(i<0||i>1)
    {
        return -1;
    }
    switch(i)
    {
        case 0:
            if(stack.top[0]!=-1)
            {
                return stack.data[stack.top[0]--];
            }
            break;

        case 1:
            if(stack.top[1]!=-stack.MAXSIZE)
            {
                return stack.data[stack.top[1]++];
            }
            break;
    }
}
}

```

代码题 4

题干

- 题目:

4、判断一个表达式中括号是否配对(假设只包含圆括号)

- 总结:

- 使用栈进行匹配，分为两个动作;
- 左括号：啥也不用管，入栈就完事了;
- 右括号：将栈顶弹出，看是否和右括号匹配。如果匹配则继续，如果不匹配就配对失败;

解答

```
bool isMatch(string str, stack &stack)
{
    int i = 0;

    while(str[i]!='\0')
    {
        switch(str[i])
        {
            case '(':
                push(stack, '(');
                break;
            case ')':
                char tmp = pop(stack);
                if(tmp=='(')
                {
                }
                else
                {
                    return false;
                }
                break;
        }
        i++;
    }

    if(stack.top == -1)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

代码题 5

题干

- 题目:

5、假设一个序列为 HSSHHS, 运用栈的知识,
• 编写算法将 S 全部提到 H 之前, 即为 SSSHHH

- 总结:

- 用两个数据结构存储两种符号;
- 如果是 S, 则加入数组;
- 如果是 H, 则加入栈;
- 遍历完序列后, 将栈中元素逐个出栈、放入 S 后;

解答

```
string fun(string str)
{
    stack s;
    Init(s);
    string newStr = "";

    // 用于遍历str
    int i = 0;
    // 用于指向newStr末尾
    int j = 0;
    while(str[i]!='\0')
    {
        if(str[i]=='S')
        {
            newStr[j++] = str[i++];
        }
        else if(str[i]=='H')
        {
            push(stack, str[i++]);
        }
    }

    while(stack.top!=-1)
    {
        newStr[j++] = pop(stack);
    }

    return newStr;
}
```