数据结构实验报告

第 5 次实验	学号：20191060064	姓名：白文强	得分：

一、实验目的

1、复习栈和队列的逻辑结构、存储结构及基本 操作；
2、掌握顺序栈、链队列；
3、了解链栈、循环队列。

二、实验内容

1、（必做题）假设栈中数据元素类型是字符型，请 采用顺序栈实现栈的以下基本操作：
（1）Status InitStack (&S)　　//构造空栈 S；
（2）Status Push(&S, e)　　//元素 e 入栈 S；
（3）Status Pop(&S, &e)　//栈 S 出栈，元素为 e。
2、（必做题）假设队列中数据元素类型是字符型， 请采用链队列实现队列的以下基本操作：
（1）Status InitQueue(&Q)　　//构造空队列 Q；
（2）Status EnQueue(&Q, e)　　//元素 e 入队列 Q；
（3）Status DeQueue (&Q, &e)　//队列 Q 出队列，元 素为 e。
3、（必做题）请实现：对于一个可能包括括号 {}、[]、()的表达式，判定其中括号是否匹配。
4.1、（选做题）数据结构 MOOC 的第三章编程实训题
　　3-1 另类循环队列（在拼题 A 网站 https://pintia.cn/上 完成）
4.2、(选做题) 数据结构 MOOC 的第三章编程实训题
　　3-4 堆 栈操作合法性（在拼题 A 网站 https://pintia.cn/上 完成）

三、数据结构及算法设计

1、栈的基本操作
2、队列的基本操作
3、利用出栈入栈匹配括号
4.1、不设队列尾指针，另设 Count 记录队列中元素个数
4.2、设一变量表示栈的长度，当长度不符合要求时，输出 NO，否则输出 YES

四、核心程序代码（给出必要注释）

一、

```
Status InitStack(sqStack *s)//初始化空栈
{
    s->base = (ElemType *)malloc(STACK_INIT_SIZE * sizeof(ElemType));
    if(!s->base){
        return -1;
    }
    else
    {
        s->top = s->base;//最开始，栈顶就是栈底
        s->stackSize = STACK_INIT_SIZE;
    }
    return 1;
}
```

```
Status Push(sqStack *s, ElemType e)//入栈
{   //如果栈满，追加空间
    if(s->top - s->base >= s->stackSize)
    {
        s->base = (ElemType *)realloc(s->base, (s->stackSize + STACK_INIT_SIZE) *
sizeof(ElemType));
        if(!s->base){
            return -1;
        }
        else{
            s->top = s->base + s->stackSize;//设置栈顶
            s->stackSize = s->stackSize + STACKINCREMENT;
        }
    }
    *(s->top) = e;
    s->top++;
    return 1;
```

```
Status Pop(sqStack *s, ElemType *e)//出栈操作
{
    if(s -> top == s->base)
    {
        return -1;
    }
    else
    {
        *e = *--(s->top);
    }
    return 1;
}
```

二、

```
Status  InitQueue(LinkQueue  *q)//初始化一个空队列
{
      q->front  =  q->rear  =  (QueuePrt)malloc(sizeof(QNode));
      if(!q->front){
            return  -1;
      }
      else
      {
            q->front->next  =  NULL;
      }
      return  1;
}
```

```
Status  EnQueue(LinkQueue  *q,  ElemType  e)//入队列操作
{
      QueuePrt  p;
      p  =  (QueuePrt)malloc(sizeof(QNode));
      if(p  ==  NULL){
            return  -1;
      }
      p->date  =  e;
      p->next  =  NULL;
      q->rear  ->next  =  p;
      q->rear  =  p;
      return  1;
}
```

```
Status  DeQueue(LinkQueue  *q,  ElemType  *e)//出队列操作
{
      QueuePrt  p;
      if(q->front  ==  q->rear){
            return  -1;
      }
      p  =  q->front->next;
      *e  =  p->date;
      q->front->next  =  p->next;
      if(q->rear  ==  p)
      {
            q->rear  =  q->front;
      }
      free(p);
      return  1;
}
```

三、

```
int IsMatch(char input[],sqStack*S)
{
    char c , e ;
    int change = 0;
    for(int i = 0 ; (c = input[i]) != '\0' ; i++)//每次读取一个字符
    {
        if(c == '(' || c == '[' || c == '{')//如果是左括号，就入栈
        {
            change = 1;
            Push(S, c);
        }
        else if(')' == c)//若是小右括号，则出栈
        {
            change = 1;
            Pop(S, &e);
            if(e != '(')//若未找到小左括号
                return 0;
        }
        else if(']' == c)//若是中右括号，则出栈
        {
            change = 1;
            Pop(S, &e);
            if(e != '[')//若未找到中左括号
                return 0;
        }
        else if( '}' == c)//若是大右括号，则出栈直到找到大左括号
        {
            change = 1;
            Pop(S, &e);
            if(e !='{' )//若栈为空时还未找到大左括号，则必定无法匹配
                return 0;
        }
        else//若不是括号，则不作操作，保证栈里只存在括号
            continue;
    }
    if(change == 0)//式子中没有括号
        return 2;
    if(StackLen(*S) != 0)//栈内不空，则必定栈里有左括号无法匹配。
    {
        return 0;
    }
    return 1;
}
```

四．1

```
bool AddQ( Queue Q, ElementType X )//入队列
{
    if(Q->Count == Q->MaxSize)//若队列已满
    {
        printf("Queue Full\n");
        return false;
    }
    else//若队列未满
    {
        Q->Data[(Q->Front + Q->Count)%Q->MaxSize] = X;//(Q->Front +
Q->Count)/Q->MaxSize  表示队尾元素的下一个元素的下标
        Q->Count++;//元素数加1
    }
    return true;
}
```

```
ElementType DeleteQ( Queue Q )//出队列
{
    ElementType e;
    if(0 == Q->Count)//队空
    {
        printf("Queue Empty\n");
        return ERROR;
    }
    else
    {
        e = Q->Data[Q->Front];//先取队首元素
        Q->Front = (Q->Front + 1)%Q->MaxSize;//头指针前移
        Q->Count--;//元素数减一
    }
    return e;
}
```

四．2

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    int N, M;
    scanf("%d%d", &N, &M);
    char Input[N][101];
    for(int i = 0; i < N; i++)
        scanf("%s", Input[i]);//读取N个字符串
    for(int i = 0 ; i < N ; i++)
    {
        int Count = 0;//表示栈的长度
        int length = strlen(Input[i]);//字符串长度
        for(int j = 0 ; j < length ; j++)
        {
            if('S' == Input[i][j])
            {
                Count++;//是S就入栈，栈的长度加一
            }
            else
            {
                Count--;//否则是X，出栈，栈的长度加一
            }
            if(Count > M || Count < 0)//如果栈的长度大于最大长度，或者小于0，则
非法
            {
                printf("NO\n");
                break;
            }
            if(length - 1 == j)//到字符串最后一个元素时
            {
                if(Count == 0)//栈空，则合法
                {
                    printf("YES\n");
                }
                else//栈非空，则非法
                {
                    printf("NO\n");
                }
            }
        }
    }
    return 0;
}
```

五、测试及结果（给出测试用例及测试结果）

一、

二、

三、

四、

五