# 中国科学技术大学计算机学院
# 《数据结构》报告


实验题目：栈、队列及其应用

学生姓名：高楚晴

学生学号：PB18111688

完成日期：2019.11.13

# 实验目的

深入理解栈和队列的特性，领会它们各自的应用背景。熟练掌握它们在不同存储结构、不同的约定中，其基本操作的实现方法与差异。实际体会顺序栈与链栈的区别、链队列与顺序队列/循环顺序队列的区别。

# 设计思路

**1.必做部分**

本次选择魔王语言解释的实验，基本思路如下。

为提高程序的扩展性（如需再对原文本进行其他特殊规定），先使用数组将魔王语言存放在一个字符数组中，同样的，为方便后续使用，将转换后的魔王语言也存入字符数组而不是直接在栈弹出时输出，

由于前后存放数组的存在，基本思路为：读取括号外元素时将小写字母直接存入目标数组，将大写字母转换后再存放；读取括号内元素时，依次压入栈中，括号结束时弹出并一一压入栈中，当遇到大写字母时做额外处理。

**2.选做部分**

下面使用队列实现，在操作方面分为两部分，对于括号外的内容直接输出，括号外的部分入队列，通过修改出队列操作使其从队尾出队列，模拟栈的实现。

# 关键代码讲解

## 1.栈实现

下面展示的是主函数部分，由于主体位于一个大的for循环中，难以拆分展示，仅在注释中予以解释。

```
for (i = 0; ori[i]; i++) {                    //读取魔王语言并转化
        if (ori[i] == 'B') {                  //对需要翻译的大写字母进行特殊约定
            strcat(aftertran, "tsaedsae");
            curlen += 8;
            continue;
        }
        if (ori[i] == 'A') {
            strcat(aftertran, "sae");
            curlen += 3;
            continue;
        }
        if(ori[i] == '('){                    //从遇到括号起开始将字符一一压入栈中
            i++;                              //跳过左括号
            head = ori[i++];                  //记录第一个字符
            pStack = CreateStack();
            while (ori[i] != ')') {           //将第一个字符以外的压入栈中
                Push(ori[i], pStack);
                i++;
            }
            while (pStack->next != NULL) {    //括号内入栈完毕，下面开始出栈并转存
                temp = Pop(pStack);
                aftertran[curlen++] = head;
                if (temp == 'A') {
```

```
                    for (j = 1; j < 4; j++)
                        aftertran[curlen++] = str[j];
                    continue;
                }
                if (temp == 'B') {
                    for (j = 0; j < 8; j++)
                        aftertran[curlen++] = str[j];
                    continue;
                }
                aftertran[curlen++] = temp;
            }
            aftertran[curlen] = head;
            free(pStack);                       //释放节点
            continue;                           //继续翻译下一个字符
        }
        aftertran[curlen++] = ori[i];           //括号外的小写字母，直接转存
}
```

### 2.队列实现

主要实现为出队列时从队尾出，以此模拟栈的功能,实现函数如下，其他思路和必做部分基本类似。

```
int DeQueuet(SqQueue &Q, char &e) {                 //删除队尾元素并返回其值
    if (Q.front == Q.rear)
        return ERROR;
    e = Q.base[Q.rear-1];
    Q.rear = (Q.rear - 1 + MAXSIZE) % MAXSIZE;
    return OK;
}
```

## 调试分析

初次完成代码后进行调试，发现运行失败，经检查为误在出栈pop操作之后再进行了p = p->next操作，调试后运行正常。

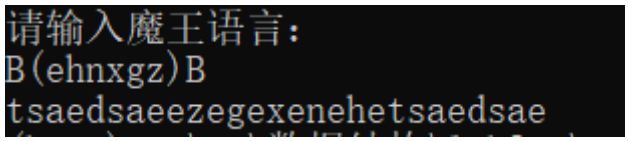
再分析算法的时空复杂度，若计魔王语言的长度为n，存放魔王语言的字符串占用空间n，翻译后的占用空间为ax+b，a，b均为常数，过程中使用的栈占空间小于n，因此空间复杂度为 $O(n)$ 。

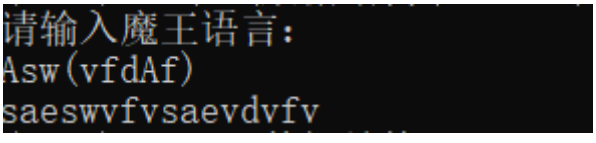
在时间复杂度上，括号内部分语言翻译时时间复杂度为字符长度的常数倍，括号外大写字母翻译时间复杂度为翻译长度的常数倍，小写字母复杂度为其长度本身，综合来看，时间复杂度为 $O(n)$。

队列实现算法同样，时空复杂度均为$O(n)$。

## 代码测试

使用实验指导中的测试案例，再随机输入一组别的数据进行测试，结果如下。



## 实验总结

通过本次实验掌握了栈和队列的基本使用，对其储存空间的形式有了更好地理解，通过试错学习强化了栈和队列的相关基本操作。

## 附录

PB18111688_高楚晴_2_1.cpp　=> 必做部分

PB18111688_高楚晴_2_2.cpp　=>选做栈实现