

# 《数据结构》

## 课程设计报告

课程名称：	《数据结构》课程设计
课程设计题目：	魔王语言
姓 名：	金宏睿
院 系：	计算机学院
专 业：	计算机科学与技术
班 级：	19052318
学 号：	19051814
指导教师：	肖周芳

2020 年 12 月 4 日

## 一、需求分析

### 功能需求：

用规则  $B \rightarrow tAdA$  和  $A \rightarrow sae$  实现一个魔王语言的解释系统，把他的话解释成人能听得懂的语言。魔王的语言是依据以下两种形式的规则由人的语言逐步抽象上去：

$$(1) \quad \alpha \rightarrow \beta_1 \beta_2 \cdots \beta_3$$

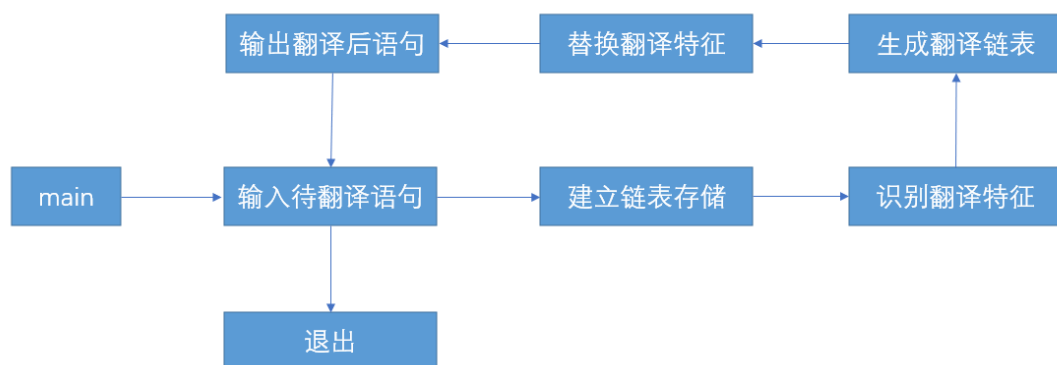
$$(2) \quad (\theta \delta_1 \delta_2 \cdots \delta_n) \rightarrow \theta \delta_n \theta \delta_{n-1} \cdots \theta \delta_1 \theta$$

设大写字母表示魔王语言的词汇；小写字母表示人的语言词汇；希腊字母表示可以用大写字母或小写字母代换的变量。魔王语言可含人的词汇。

### 界面需求：

需要有提示引导使用者输入待翻译的语句，输入完成后，输出翻译后的语句

## 二、概要设计



### 模块层次结构设计

`void TransHome()` //翻译功能的入口

### 接口设计

//翻译语言中的大写字母  
`void ReplaceUP(LNode *qu)`

```

//替换字符串中的括号内部的字符串
void ReplaceBR(LinkList qu)

//在遍历过程中决定是否对字符进行替换
LNode *Replace(LNode *tp)

//创建字符串并进行处理
void TransHome()

```

## 数据结构设计

```

typedef struct LNode
{
    ElementType data;
    struct LNode *next;
    struct LNode *pre;
} * LinkList; //链表及结点结构体

LinkList words; //存储翻译的语言的链表

typedef struct StNode
{
    StNode() {next = NULL;}
    ElementType data;
    StNode *next;
    StNode *pre;
} * Stack; //栈及栈结点结构体

```

## 三、详细设计

```

#include <bits/stdc++.h>
using namespace std;
using ElementType = char;

//链表及结点结构体，用于存储翻译的语言
typedef struct LNode
{
    ElementType data;
    struct LNode *next;
    struct LNode *pre;
} * LinkList;
LinkList words;

//从输入获取字符并创建链表存储

```

```

LinkedList LLCreat()
{
    LinkedList ll = new LNode;
    ElementType dt;
    LNode* re = ll;
    while(dt = getchar(),dt != '\n')
    {
        LNode *tp = new LNode;
        tp->data = dt;
        tp->pre = re;
        re->next = tp;
        re = re->next;
    }
    re->next = new LNode;
    re->next->next = NULL;
    return ll;
}

```

//栈及栈结点结构体

```

typedef struct StNode
{
    StNode() {next = NULL;}
    ElementType data;
    StNode *next;
    StNode *pre;
} * Stack;

```

//根据坐标将结点插入链表

```

void LNodeInsert(LinkedList ll, int pos,ElementType dt)
{
    LNode *re = ll->next;
    for(int i = 1;i<pos;i++)
    {
        re = re->next;
    }
    LNode *tp = new LNode;
    if(re->next == NULL)
    {
        tp->next = NULL;
        tp->pre = re;
        tp->data = dt;
        re->next = tp;
    }
}

```

```

        else{
            tp->data = dt;
            tp->pre = re;
            tp->next = re->next;
            re->next->pre = tp;
            re->next = tp;
        }
    }
}

```

//出栈

```

ElementType StPop(Stack st)
{
    ElementType dt;
    if (st->next)
    {
        StNode *ttp = st;
        StNode *tp = st->next;
        while(tp->next)
        {
            tp = tp->next;
            ttp = ttp->next;
        }
        dt = tp->data;
        ttp->next = NULL;
        delete tp;
    }
    else
    {
        return 0;
    }
    return dt;
}

```

//入栈

```

void StPush(Stack st, ElementType data)
{
    StNode *tp = st;
    if (st->next)
    {
        while (tp->next)
            tp = tp->next;
        StNode *ttp = new StNode;
        ttp->data = data;
        tp->next = ttp;
    }
}

```

```

        ttp->pre = tp;
        ttp->next = NULL;
    }
    else
    {
        tp = new StNode;
        tp->data = data;
        st->next = tp;
        tp->pre = st;
        tp->next = NULL;
    }
}

```

//栈的初始化

```

Stack StInit(Stack st)
{
    StNode *sn = st->next;
    if (!st)
    {
        st = new StNode;
        st->next = NULL;
    }
    else
    {
        StNode *tp = st->next;
        while (tp)
        {
            sn = tp;
            tp = tp->next;
            delete sn;
        }
        st->next=NULL;
    }
    return st;
}

```

//判断栈是否为空

```

bool StEmpty(Stack st)
{
    if(st->next)
        return 0;
    return 1;
}

```

```
//重载操作符便于输出链表的数据
ostream& operator << (ostream& os , LinkList ll)
```

```
{
    LNode *tp = ll->next;
    while(tp->next)
    {
        os << tp->data ;
        tp = tp->next;
    }
    return os;
}
```

```
//翻译语言中的大写字母
```

```
void ReplaceUP(LNode *qu)
```

```
{
    //获取当前字母的前后字母的指针
    LNode *qunext = qu->next;
    LNode *qupre = qu->pre;

    //字母 A 和 B 分别替换，用替换之后的链表替换原来的字母结点
    if (qu->data == 'A')
    {
        LNodeInsert(qupre, 1, 's');
        LNodeInsert(qupre, 2, 'a');
        LNodeInsert(qupre, 3, 'e');
        qupre->next = qu->next;
        qu->next->pre = qupre;
        delete qu;
    }
    else if (qu->data == 'B')
    {
        LNodeInsert(qupre, 1, 't');
        LNodeInsert(qupre, 2, 'A');
        LNodeInsert(qupre, 3, 'd');
        LNodeInsert(qupre, 4, 'A');
        qupre->next = qu->next;
        qu->next->pre = qupre;
        delete qu;
    }
}
```

```
//替换字符串中的括号内部的字符串
```

```
void ReplaceBR(LinkList qu)
```

```
{
```

```

//获取当前字符前字符的指针
LNode *qupre = qu->pre, *qunext;
//遇到括号就处理
if (qu->data == '(')
{
    LNode *tp = qu;          //保存括号指针
    Stack st = new StNode;    //开辟栈用于存储首字符之后的字符
    st = StInit(st);          //初始化栈
    if (qu->next->data == ')') //如果括号内为空，则清空括号，结束函数
    {
        tp->pre->next = tp->next->next;
        tp->next->next->pre = tp->pre;
        delete tp;
        return;
    }
    ReplaceUP(qu->next);      //处理首字符为大写字母的情况
    LNode seta = *qu->next;    //保存首字母的结点的值
    tp = qu->next->next;       //临时指针指向第二个字符
    tp->pre = qupre;           //第二个字符指向括号前一字符
    LNode *de = qu->next;      //指向首字符
    delete qu;                //释放括号结点的空间

    //遍历找到后半括号,遍历过程中将字符结点入栈
    while (tp->data != ')')
    {
        LNode *de = tp;      //de 用于 tp 遍历到下一结点后释放空间
        StPush(st, tp->data); //字符结点入栈
        tp->pre->next = tp->next; //重新连接前驱
        tp->next->pre = tp->pre;  //重新连接后置
        tp = tp->next;          //往后位移
        delete de;              //释放空间
    }
    int count = 1;
    while (!StEmpty(st)) //字符持续出栈，并与首字符一起插入链表
    {
        LNodeInsert(qupre, count, seta.data);
        LNodeInsert(qupre, count + 1, StPop(st));
        count += 2;
    }
    LNodeInsert(qupre, count, seta.data); //插入最后一个首字母
    qupre->next = tp->next;
    tp->next->pre = qupre;
    delete tp;
}

```



```
}
```

```
//在遍历过程中决定是否对字符进行替换
```

```
LNode *Replace(LNode *tp)
```

```
{
```

```
    ReplaceUP(tp->next);
```

```
    ReplaceBR(tp->next);
```

```
    return tp;
```

```
}
```

```
//创建字符串并进行处理
```

```
void TransHome()
```

```
{
```

```
    cout << "words before translate : ";
```

```
    words = LLCreat();
```

```
    LNode *tp = words;
```

```
    while (1)
```

```
    {
```

```
        tp = Replace(tp);
```

```
        tp = tp->next;
```

```
        if (!tp->next)
```

```
            break;
```

```
    }
```

```
}
```

```
int main()
```

```
{
```

```
    while (1)
```

```
    {
```

```
        TransHome();
```

```
        cout << endl;
```

```
        cout << "words atfer translate : ";
```

```
        cout << words << endl;
```

```
        cout << endl;
```

```
        cout << endl;
```

```
    }
```

```
    return 0;
```

```
}
```

## 四、调试分析

- 1、本次作业要求设计一个语言翻译程序，要求能根据翻译的约束规则，对输入的相应语句进行翻译。
- 2、本程序的一个难点在于语句中括号和大写字母的交替时的处理，为了方便处理，我们采用链表对语句进行存储，并且优先级采用大写字母优先。
- 3、在处理括号内的语句时，由于需要逆序处理，很符合栈的特征，所以采用栈进行中间存储。

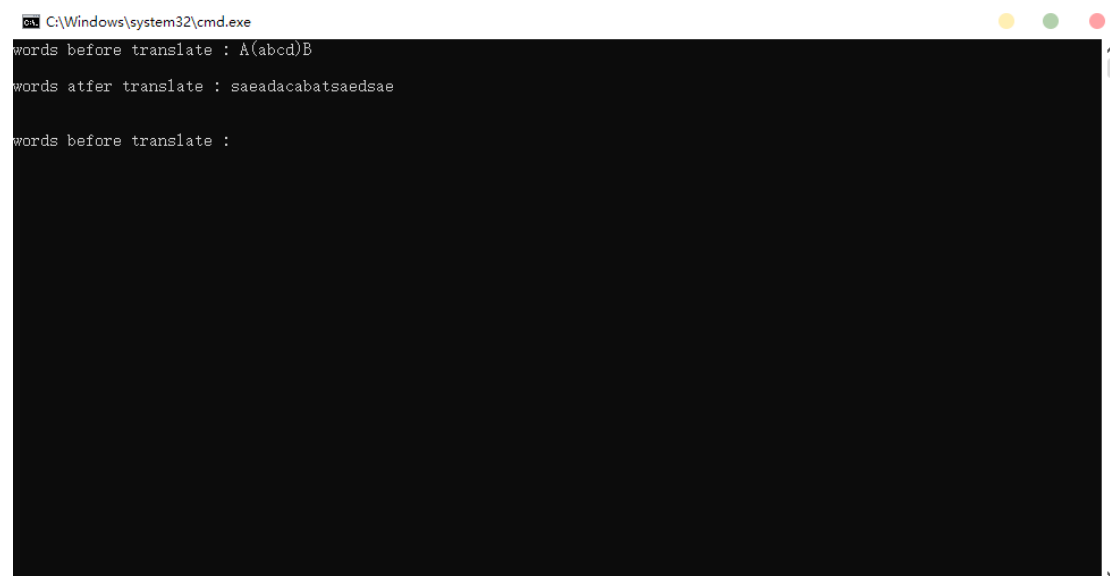
## 五、用户手册

- 1、本程序的执行文件为：devlan.exe
- 2、进入演示程序后，将显示如下的界面



A screenshot of a Windows command prompt window. The title bar shows the path "C:\Windows\system32\cmd.exe". The command prompt displays the text "words before translate : " followed by a cursor, indicating it is ready for input.

- 3、输入待翻译语句，回车确认，可得到翻译后的结果。



A screenshot of a Windows command prompt window showing the program's output. The title bar shows the path "C:\Windows\system32\cmd.exe". The command prompt displays the following text:  
words before translate : A(abcd)B  
words atfer translate : saeadacabatsaedsae  
words before translate :  
The output shows the original input "A(abcd)B" and the translated result "saeadacabatsaedsae". The word "atfer" is misspelled as "after".

- 4、随后可选择进入下一次翻译后者自行退出程序。

## 六、测试结果

对于不同的输入，都能获得相应的结果：

```
C:\Windows\system32\cmd.exe
words before translate : a(aA)b
words after translate : aasaeab

words before translate : a(Ab)b
words after translate : asbsesab

words before translate : A(AB)B
words after translate : saestsaeasasestsaeasae

words before translate : (abcd)
words after translate : adacaba

words before translate : A(abcd)B
words after translate : saeadacabatsaeasae

words before translate :
```

输入：

a(aA)b  
a(Ab)b  
A(AB)B  
(abcd)  
A(abcd)B

## 七、附录

源程序文件名清单：devlan.cpp，devlan.exe