**魔王语言解释**

**一 目的**

编写一个魔王语言解释程序，使其能够按一定规则和顺序来翻译成人类语言，要求利用栈的先进后出和队列的先出后进的特性来处理，利用栈和队列的特性来处理魔王语言与人类语言之间的对应关系，然后用数组将翻译结果存储以便翻译成中文和写入文件。

**二 需求分析**

**1、设计魔王语言解释规则**

根据任务书的要求，我们应该先设定或者存储自定义一些魔王语言解释规则，魔王语言解释主要通过栈来完成。

**2、运行魔王语言解释**

1）根据设定或自定义的解释规则，自定义的规则在文件rules写入或载入。

2）输入需要翻译的魔王语言，魔王语言经过翻译后选择将魔王语言与翻译后的语言存入trans文件

**三 概要设计**

**1、主程序模块**

main()

{

定义及初始化；

打印用户选择菜单，等待用户输入；

执行指定规则的魔王语言翻译；

}

**2、实现可翻译的抽象数据模块**

void tempStack(sqstack \*temp)

temp是处理括号内需要翻译语言的数据结构栈

操作结果：遇到’(’后将后续的字符串保存到栈temp，直到遇到‘)’后停止入栈

void spenqueue(linkqueue \*queue, char key,rules rule) 入队函数

queue是用于保存翻译为最终形态的魔王语言的数据结构队列

key 是魔王语言的一个待翻译字符

rule 是用户选在载入的翻译规则

操作结果：将每次输入的字符按照用户指定的规则翻译，结果push到队列queue中

status sort(sqstack \*stack, linkqueue \*queue,rules rule)

stack 是用于保存的括号内的魔王语言的数据结构栈

queue 是用于保存最终翻译结果的数据结构队列

rule 是用户选择的的翻译规则

操作结果：将魔王语言翻译，保存结果到queue

void print(char word[100])

word 是翻译后的魔王语言

操作结果：返回值为魔王语言与汉字的一一对应关系的汉语

rules choicerule()

操作结果：读取文件rules ，返回所选择的翻译规则

**四 详细设计**

**1、用于处理括号内语言的栈**

typedef struct

{

selemtype \*base;

selemtype \*top;

int stacksize;

} sqstack;

**2、初始化栈**

status initStack(sqstack \*stack)

{

stack->base = (selemtype \*) malloc (stack\_init\_size\*sizeof(selemtype));

if(!stack->base) exit(OVERFLOW);

stack->top = stack->base;

stack->stacksize = stack\_init\_size;

return 1;

}

**3、对栈的基础操作，进栈**

status push(sqstack \*stack, selemtype e)

{

if(stack->top - stack->base == stack->stacksize)

{

stack->base =(selemtype \*) realloc(stack->base,(stack->stacksize+stackincrement)\*sizeof(selemtype));

if(!stack->base) exit(OVERFLOW);

stack->top = stack->base + stack->stacksize;

stack->stacksize+=stackincrement;

}

\*(stack->top++)=e;

return 1;

}

**4、队列结构（链表）**

typedef struct qNode{

qelemtype data;

struct qNode \*next;

}qNode,\*queueptr;

typedef struct

{

queueptr front;

queueptr rear;

}linkqueue;

**5、队列基础操作，初始化队列**

status initQueue(linkqueue \*queue)

{

queue->front = queue->rear = (queueptr)malloc(sizeof(qNode));

if(!(queue->front)) exit(OVERFLOW);

queue->front->next = NULL;

}

**6、队列基础操作，入队**

status qPush(linkqueue \*queue, qelemtype e)

{

if(!queue->front) exit(OVERFLOW);

queueptr p;

p = (queueptr)malloc(sizeof(qNode));

if(!p) exit(OVERFLOW);

p->data = e;

p->next = NULL;

queue->rear->next = p;

queue->rear=p;

return 1;

}

**7) 队列基础操作，出队**

status qPop(linkqueue \*queue,qelemtype \*e)

{

queueptr p;

if(queue->front == queue->rear) return -1;

p = queue->front->next;

\*e = p->data;

queue->front->next = p->next;

if(queue->rear == p) queue->rear = queue->front;

free(p);

return 1;

}

伪代码

void tempStack(sqstack \*temp)

{

for（从魔王语言中读入字符，遇到“#“停止）

遇到“（”将后续字符压入栈temp中直到”）”结束。

}

void spenqueue(linkqueue \*queue, char key,rules rule)

{

读入字符key。

如果符合规则1将该字符替换为1的翻译结果。

如果符合规则2将该字符替换为2的翻译结果。

将翻译结果push进入queue中

}

status sort(sqstack \*stack, linkqueue \*queue,rules rule)

{

读入魔王语言。

如果该字符在 a~z范围之间，将其push进入queue

否则再判断 该字符是否再 A~Z之间，如果是 调用sequeue函数将其转化push进入queue中，如果该字符为“（“，将其后入内容push进入queue中，直到遇到”）“

}

void print(char word[100])

{读入翻译后的魔王语言，将字符与汉字进行一一对应打印出来。并将魔王语言，翻译后的语言，和汉语写入文件trans中

}

rules choicerule()

{

打开文件rules 载入所有规则，并打印到屏幕上

将其规则保存到结构体rule并返回

}

main()

{

|  |
| --- |
|  |
|  |

打印选择菜单

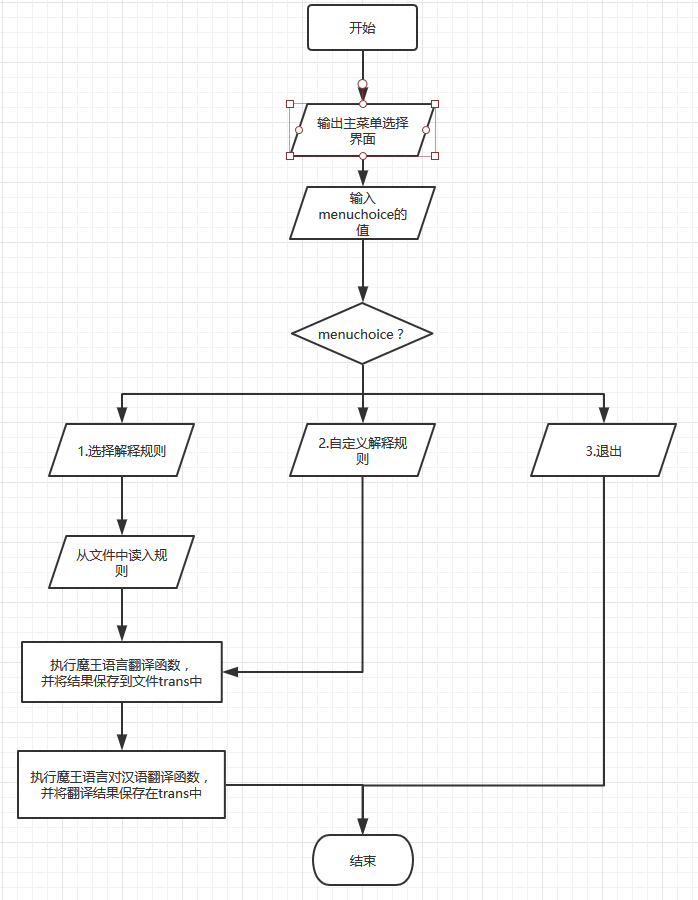
用户选择

执行对应的翻译规则，并等待用户输入魔王语言。

将魔王语言翻译为人话，并将翻译后的结果保存到文件trans中。

}

整个程序的流程图如下：

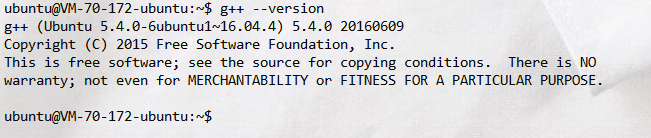
执行对应的翻译规则，并等待用户输入魔王语言。执行对应的翻译规则，并等待用户输入魔王语言。

**五 调试分析**

1. 运行环境

运行系统为ubuntu 16.04 LTS 64bit

g++版本如下



1. 该程序的关键是正确使用栈与队列分别对括号中元素保存以及翻译结果入队，这样就可以保证翻译结果的顺序正确以及对完整保存括号中的元素。
2. 对不同翻译规则和自定义翻译规则有较好的典型性和可比性。
3. 本程序遵循增量编程的策略，在实行翻译规则可自定义之前，先完成实验任务书的示例规则的翻译，在测试数据通过后追加规则自定义功能。
4. 控制文件的读入和读出的格式，选择合适的文件写入模式可以将程序更使用。

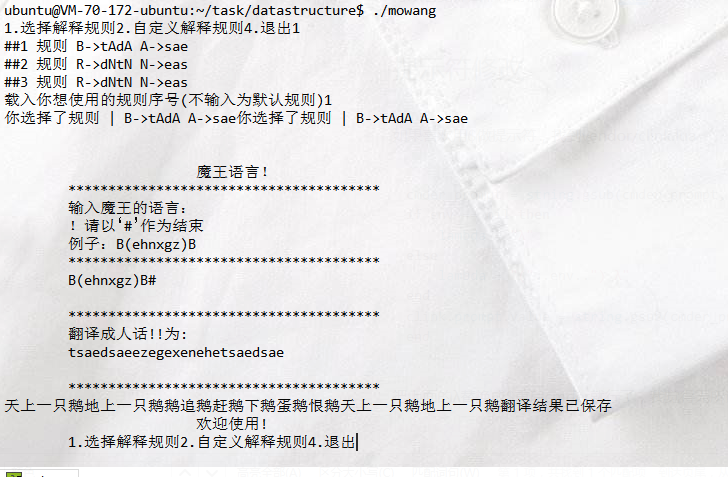
**六 测试结果**

（1）通过写该程序，熟练定义栈与队列是基本操作，合理使用两个数据结构的特性。

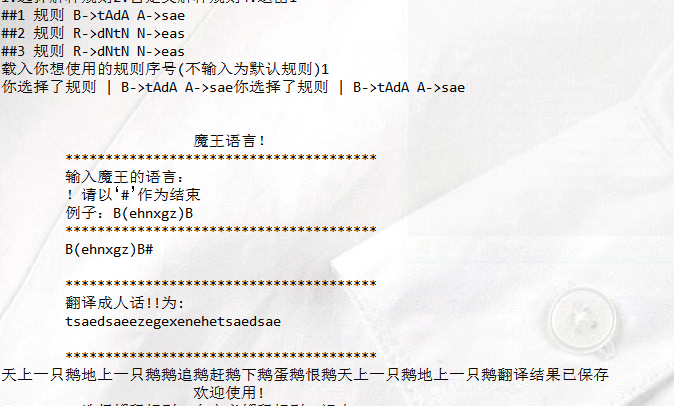
（2）主函数调用自定义规则函数，需要将输入的规则返回到主函数。因此需要使用存储规则的结构体，来传递返回值。

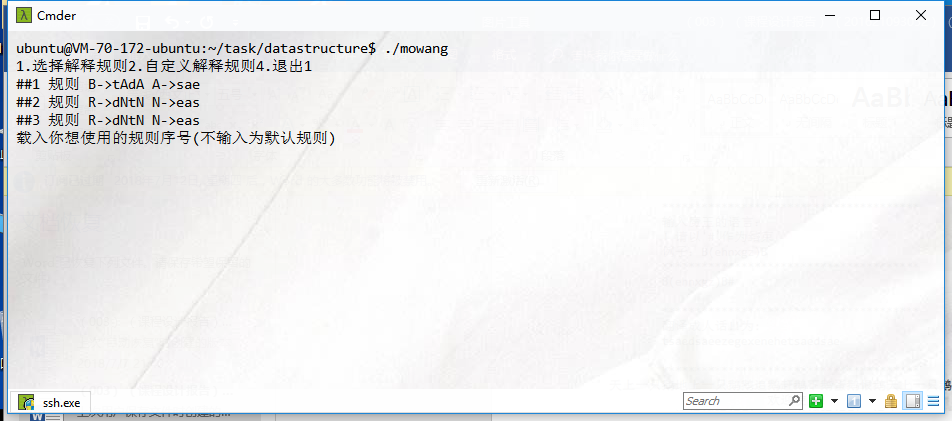
（3）打印文件内的规则需要使用feof判断是否到达文件底部，在完成文件读取后需要关闭。

（4）下图是从文件中读取一个规则后进行翻译的画面。



（5）下图是自定义一个规则后进行翻译的画面。

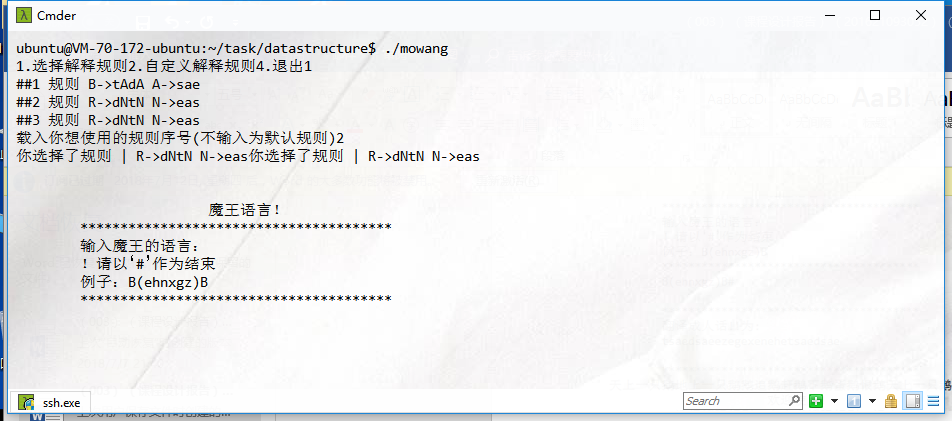


**七 用户使用说明**

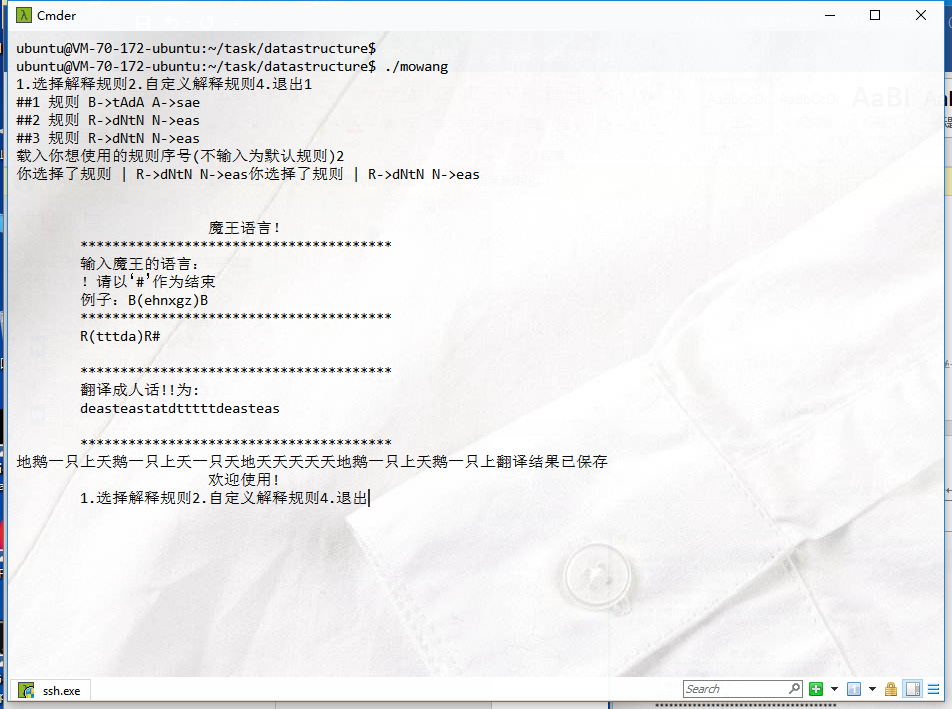
1. **选择解释规则**

用户载入已保存的翻译规则，

菜单形式提示所有可以使用的解释规则，用户选择列表中的规则。

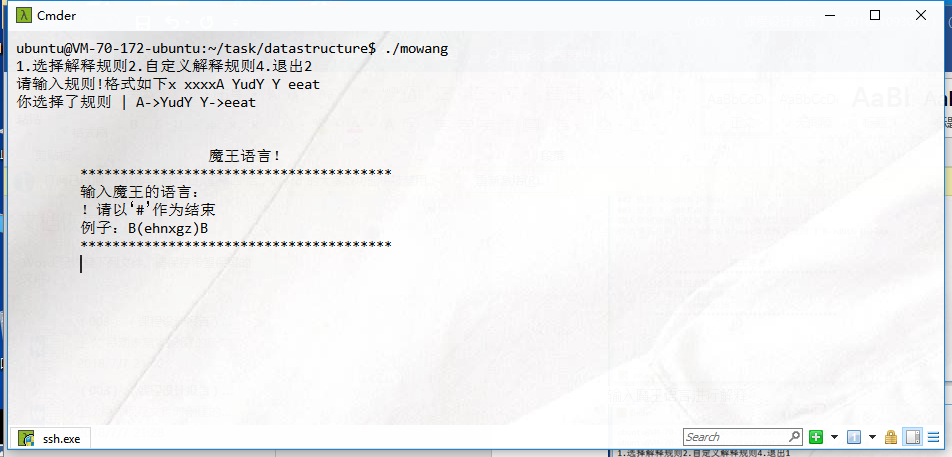


输入魔王语言进行解释

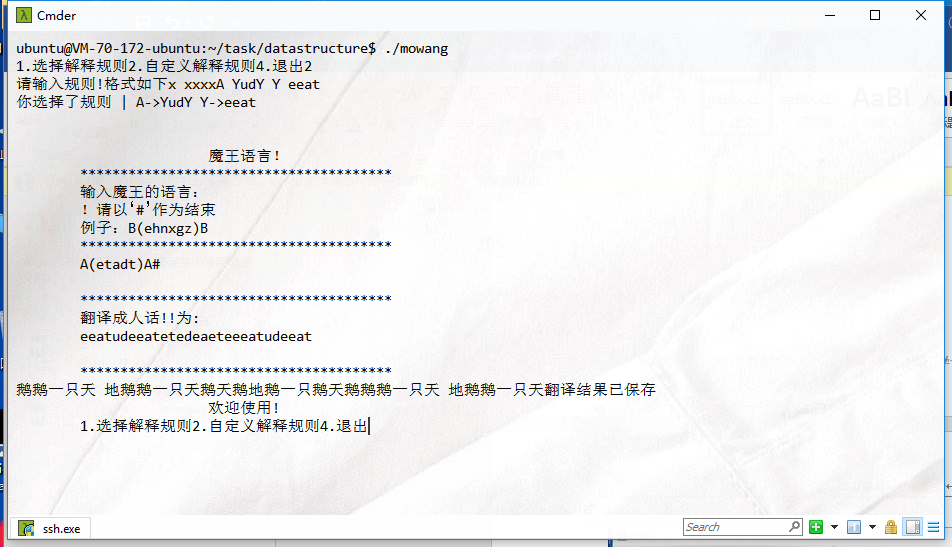


1. **自定义解释规则**

用户输入翻译规则 按照格式 A1->a1 B1->b1进行输入。

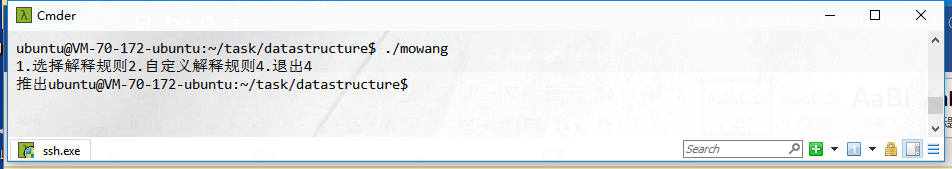


输入魔王语言进行解释



1. **选择退出程序**

在菜单下输入4 按回车退出程序。



**八 课程设计总结**

**1、栈与队列的收获总结**

除了熟练栈和队列的基本操作外，还会了一些其他的文件操作，将栈的基本功能学得更为透侧，例如：将栈与队列合起来，处理字符串。

由于队列空间分配的灵活性，可以轻易避免使用数组实现中空间浪费的问题。当然，也同时带来了其他问题，如每个元素需要额外的空间存储指针，增加了内存释放问题等。在数据量大且不确定的情况下，使用链表实现更为适合。