## 实验三 栈和队列

1. **实验目的**
   1. 了解栈和队列的特性，以便灵活应用。
   2. 熟练掌握栈和有关队列的各种操作和应用。

**二、实验内容**

**栈的基本运算：**初始化栈；进栈；退栈；判断栈是否为空栈；获取栈顶元素；输出栈的所有元素；

**队列的基本运算：**初始化队列；入队列；出队列；判断队列是否为空；判断队列头元素；

1、利用链表模拟栈实现将十进制5678转换为对应的八进制数

#include<stdio.h>

#include<stdlib.h>

#include <string.h>

typedef char DataType;

typedef struct node

{

DataType data;

struct node \*next;

}LStackNode,\*LinkStack;

void InitStack(LinkStack \*top)

/\*将链栈初始化为空。动态生成头结点，并将头结点的指针域置为空。\*/

{

if((\*top=(LinkStack)malloc(sizeof(LStackNode)))==NULL) /\*为头结点分配一个存储空间\*/

exit(-1);

(\*top)->next=NULL; /\*将链栈的头结点指针域置为空\*/

}

int StackEmpty(LinkStack top)

/\*判断链栈是否为空，就是通过判断头结点的指针域是否为空\*/

{

if(top->next==NULL) /\*判断链栈头结点的指针域是否为空\*/

return 1; /\*当链栈为空时，返回1；否则返回0\*/

else

return 0;

}

int PushStack(LinkStack top, DataType e)

/\*进栈操作就是要在链表的第一个结点前插入一个新结点，进栈成功返回1\*/

{

LStackNode \*p; /\*定义指向第i个元素的前驱结点指针pre，指针p指向新生成的结点\*/

if((p=(LStackNode\*)malloc(sizeof(LStackNode)))==NULL)

{

printf("内存分配失败!");

exit(-1);

}

p->data=e; /\*指针p指向头结点\*/

p->next=top->next;

top->next=p;

return 1;

}

int PopStack(LinkStack top,DataType \*e)

/\*删除单链表中的第i个位置的结点。删除成功返回1，失败返回0\*/

{

LStackNode \*p;

p=top->next;

if(!p) /\*判断链栈是否为空\*/

{

printf("栈已空");

return 0;

}

top->next=p->next; /\*将栈顶结点与链表断开，即出栈\*/

\*e=p->data; /\*将出栈元素赋值给e\*/

free(p); /\*释放p指向的结点\*/

return 1;

}

int StackLength(LinkStack top)

{

LStackNode \*p;

int count=0;

p=top;

while(p->next!=NULL)

{

p=p->next;

count++;

}

return count;

}

void DestroyStack(LinkStack top)

{

LStackNode \*p,\*q;

p=top;

while(!p)

{

q=p;

p=p->next;

free(q);

}

}

int GetTop(LinkStack top,DataType \*e)

{

LStackNode \*p;

p=top->next;

if(!p) /\*判断链栈是否为空\*/

{

printf("栈已空");

return 0;

}

\*e=p->data; /\*将出栈元素赋值给e\*/

return 1;

}

void main()

{

}

2、设单链表中存放n 个字符,设计一个算法,使用栈判断该字符串是否中心对称,如abccba即为中心对称字符串. （根据题目填空完善程序）

**提示：**先用create()函数从用户输入的字符串创建相应的单链表，然后调用bj()函数判断是否为中心对称字符串。在 bj（）函数中先将字符串进栈，然后将栈中的字符逐个与单链表中字符进行比较。

# include <stdio.h>

# include <malloc.h>

# define MaxLen 100

typedef struct node

{

char data；

struct node \*next；

}cnode；

cnode \*create (char s[])

{

int I=0；

cnode \*h, \*p, \*r；

while (s[I]!=’\0’)

{

p=(cnode \*)malloc(sizeof(cnode))；

p->data=s[I]；p->next=NULL；

if (I= =0)

{

h=p；

. ； /\*r始终指向最后一个结点\*/

}

else

{

r->next=p；r=p；

}

I++；

}

return h；

}

int judge(cnode \*h)

{

char st[MaxLen]；

int top=0；

cnode \*p=h；

while (p!=NULL)

{

st[top]=p->data；

top++；

p=p->next；

}

p=h；

while (p!=NULL)

{

top--；

if (p->data= =st[top])

p=p->next；

else

break；

}

if (p= =NULL)

return ；

else

return ；

}

void main()

{

char str[maxlen]；

cnode \*h；

printf（“输入一个字符串：”）；

scanf（“%c”,str）；

h=create( )；

if (judge(h)= = 1)

printf( “ str是中心对称字符串\n”）；

else

printf（“str不是中心对称字符串\n”）；

}

输入一个字符串：abccba

输出：

3、假设以数组sequ[0..MaxSize-1]存放环形队列的元素，同时设变量rear和len分别指示环形队列中队尾元素的位置和内含元素的个数。试写出此环形队列队满的条件，并设计相应入队和出队的算法。（根据题目填空完善程序）

**提示：**该环形队列对满的条件为：len= =MaxSize，队空的条件为：len= =0。由rear,len求队列头指针front的过程为：

front=rear-len+1；

if (front<0) front=front+MaxSize；

即 front=(rear-len+1+MaxSize)%MaxSize

# include <stdio.h>

# define maxsize 6

typedef char queue [maxsize]；

int rear=0, len=0；

int enqueue (queue qu, char x)

{

if (len= =maxsize)

return 0；

else

{

rear=(rear+1) %maxsize；

qu[rear]=x；

len++；

return 1；

}

}

int dequeue (queue qu, char \*x)

{

int front；

if (len= =0)

return 0；

else

{

front=(rear-len+1+maxsize) %maxsize；

\*x=qu[front]；

len--；

return 1；

}

}

void main()

{

char x；

queue qu；

printf( “a入队\n”）；

enqueue (qu, ‘a’)；

printf( “b入队\n”）；

enqueue (qu, ‘b’)；

printf( “c入队\n”）；

enqueue (qu, ‘c’)；

printf( “出队一次：”）；

dequeue (qu, &x)；

printf( “%c\n”,x)；

printf( “d入队\n”）；

enqueue (qu, ‘d’)；

printf( “e入队\n”）；

enqueue (qu, ‘e’)；

printf( “出队一次：”）；

dequeue (qu, &x)；

printf（“%c\n”,x)；

printf( “f入队\n”）；

enqueue (qu, ‘f’)；

printf( “g入队\n”）；

enqueue (qu, ‘g’)；

printf( “出队一次：”）；

dequeue (qu, &x)；

printf（“%c\n”,x)；

printf（“余下元素出列：”）；

while (len>0)

{

dequeue (qu, &x)；

printf（“%c\n”,x)；

}

printf（“\n”）；

}

输出：

. 入队

. 入队

出队一次：

. 入队

. 入队

出队一次：

. 入队

. 入队

出队一次：

余下元素出列：