**上机报告（三）**

班级：软工11601

姓名：赵惹儿

学号：201603659

序号：37

**一．源文件：Mystack.cpp**

#include "stdafx.h"

#include"Mystack.h"

#include<iostream>

using namespace std;

Status Initstack(stack&s)

{

s = new LNode;

s->next = 0;

return OK;

}

Status Destorystack(stack&s)

{

Clearstack(s);

delete s;

s = 0;

return OK;

}

void Clearstack(stack&s)

{

ElemType e;

while (Pop(s, e) == OK);

}

Status StackEmpty(stack s)

{

if (s->next == 0);

return TRUE;

return FALSE;

}

Status GetTop(stack s, ElemType &e)

{

if (s->next == 0)

return ERROR;

e = s->next->data;

return OK;

}

Status Push(stack &s, ElemType e)

{

LNode\*p = new LNode;

p->data = e;

p->next = s->next;

s->next = p;

return OK;

}

Status Pop(stack &s, ElemType &e)

{

if (s->next == 0)

return ERROR;

LNode\*p = s->next;

e = p->data;

s->next = p->next;

delete p;

return OK;

}

**二．测试代码：testMystack.cpp**

#include"stdafx.h"

#include"Mystack.h"

#include<iostream>

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

char\*p = "ABCDEF";

int n = 0x89ab8ce;

ElemType e=0;

stack s;

Initstack(s);

cout << "Push:" << n <<endl;

while (n)

{

Push(s, n % 16);

n /=16;

}

cout << "Pop:" << e << endl;

while (StackEmpty(s) == FALSE);

{

Pop(s, n);

if (n < 10)

cout << n<<endl;

else

cout << p[n - 10] << endl;

}

Clearstack(s);

cout << OK << endl;

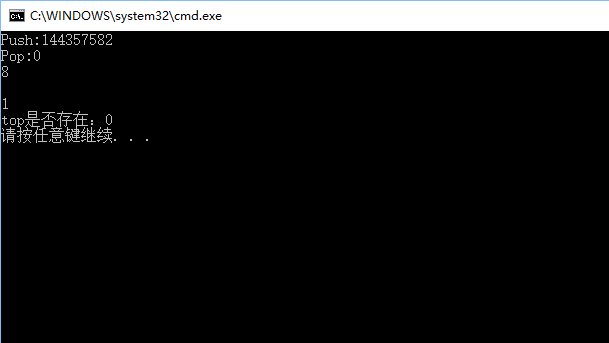
cout << "top是否存在：" << GetTop(s, e) << endl;

Destorystack(s);

return 0;

}

**三．测试结果：**



**四．测试过程分析：**

栈的表示和实现，和线性表类似。顺序栈，即栈的顺序存储结构是利用一组地址连续的存储单元依次存放自栈栈低到栈顶的数据元素，同时附设指针top指示栈顶元素在顺序栈中的位置，以top=0表示空栈。

在实现的过程中，（1）先构造了一个空栈S （Status Initstack(stack&s)），然后在空栈S中插入（Push）e为新的栈顶元素，输出插入的元素（cout << "Push:" << n <<endl;）；

（2）若栈不空，则删除S的栈顶元素，用e返回其值，并返回OK;否则返回ERROR;输出弹出（Pop）的元素；

（3）判断S是否为空栈，若S为空栈，则返回TRUE,否则返回FALSE;然后看其弹出的元素个数是否小于10，若是输出弹出的元素个数n，否则输出P[n-10];

{

Pop(s, n);

if (n < 10)

cout << n<<endl;

else

cout << p[n - 10] << endl;

}

(4)将S置为空栈，（Clearstack(s);）当S为空，返回其值OK,

（5）GetTop(s ,e) //若栈不空，则用e返回S的栈顶元素，并返回OK,否则返回ERROR,输出结果；

（6）Destorystack(s)//销毁栈S,S不在存在。

**五．总结;**

经过这几次的不断编写代码，测试代码，慢慢掌握了实现一个算法需要自己耐心的去理解，编写代码，以及多次的测试运行才能在此过程中学会一些不容易理解的知识点。学习程序更要循序渐进，一点一点积累经验。以后会尽量动手操作，实现代码。