数 据 结 构 实 验 报 告 成绩_____

学号	1307402068	姓名	李世旺	授课教师	黄 欣
专业	信息与计算科学		实验报告递交日期		2015.04.09

实验题目

两个栈共享向量空间，它们的栈底分别设在向量的两端，编制程序完成公用栈操作.

一. 需求分析

1．程序实现的功能：共用栈的各种运算。

2. 编制函数：

1). 元素进栈 i 函数 void push(seqstack *S,int i,datatype x)

2). 栈 i 退栈，返回栈顶 datatype pop(seqstack *S,int i)

3).取栈 i 顶元素 datatype top(seqstack *S,int i)

4). 主函数完成功能：

a).开辟栈空间

b).两个栈分别进栈若干个元素

c).两个栈分别退栈或取栈顶若干个元素

d). 显示最后栈内容

3．数据输入的内容、输入形式与范围

以字符串形式输入要进栈的数据，以回车符结束。

4．数据输出的内容与形式

列栈时，元素间有空格。

二. 主要算法的算法思想.

1．进栈函数：

从两端开始进栈。

2. 退栈函数：

向两端退栈。

3. 取栈顶函数：

取栈顶 top0,top1

4. 列栈函数:

从栈顶用活动的局部变量依次取元素，直到栈底

5. 主函数:

a）申请空间

b）进栈，退栈，取栈顶，列栈

c) 归还空间

三. 设计：

1．线性表存储结构：单链表。

单链表结点类型定义：

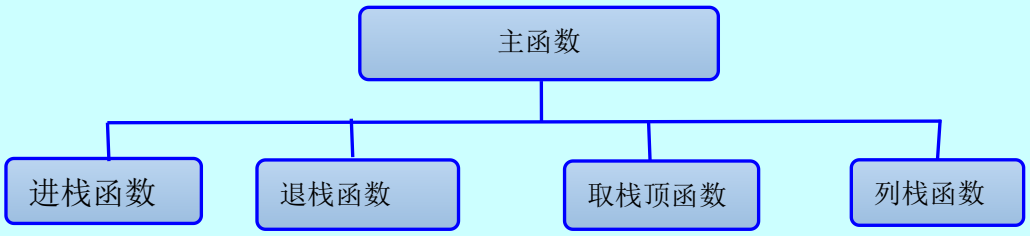
typedef struct

{ datatype V[M];int top0,top1;}seqstack;/*共用栈类型*/

2．参数表（列出所有的符号常量与全局变量）

参数名	数据传递方式	数据内容	传递	所属函数
NULL	符号常量	空指针	0	所有函数

3．函数间的调用关系图



4．列出每个函数的函数声明、函数作用、函数值、形参内容与形式、主要算法步骤等

1)．进栈函数

函数首部：void push(seqstack *S,int i,datatype x)

形参：S：共用栈类型指针；i:栈 0 或栈 1；x:进栈元素

函数作用：进栈

函数值：无

局部变量　无

算法主要步骤：

(a) 判断栈满了没　　if(S->top0==S->top1){printf("overflow");}

(b) 进栈相关操　　else { if(i==0){S->top0++;S->V[S->top0]=x;}

if(i==1){S->top1--;S->V[S->top1]=x;}

}

2)．退栈函数

函数首部：datatype pop(seqstack *S,int i)

形参：S：共用栈类型指针；i:栈 0 或栈 1；

函数作用：退栈

函数值：退栈元素

局部变量　无

算法主要步骤：

(a)先判空，再退栈

```
if(i==0)
{if(S->top0<=-1){printf("\nstack0underflow\n");return NULL;}
else {S->top0--;return S->V[++(S->top0)];}
}
if(i==1)
{if(S->top1==M){printf("\nstack1underflow\n");return NULL;}
else {S->top1++;return S->V[--(S->top1)];}
}
```

3). 取栈顶函数

函数首部： datatype top(seqstack *S,int i)

形参：S：共用栈类型指针；i:栈 0 或栈 1；

函数作用：取栈顶

函数值：栈顶元素

局部变量：无

算法主要步骤：

(a) 先判空，再取栈顶

```
if(i==0)
{if(S->top0<=-1){printf("\nstack0empty\n");return NULL;}
else {S->top0--;return S->V[(S->top0)];}
}
                if(i==1)
{if(S->top1==M){printf("\nstack1empty\n");return NULL;}
else {S->top1++;return S->V[(S->top1)];}
}
```

4）. 列栈函数：

函数首部：void showstack(seqstack *S)

形参：S：共用栈类型指针；

函数作用：列出栈元素

函数值：无

局部变量：r1：从栈顶逐渐指向栈底；

r2：从栈顶逐渐指向栈底；

算法主要步骤：

(a) 用活动的局部变量依次取栈元素

```
int r1=S->top0,r2=S->top1;
    printf("\nshow stack0\n");
while(r1>=0){printf("%c ",S->V[r1]);r1--;}
    printf("\nfinish stack0\n");
    printf("\nshow stack1\n");
while(r2<M){printf("%c ",S->V[r2]);r2++;}
    printf("\nfinish stack1\n");
```

四. 调试分析：

1．调试中出现的问题，解决的办法

1）．含有//的行出现错误：改为/**/格式

2）．出现大量错误：形如 typedef char datatype 加上分号

2．每个函数的时、空复杂性分析

1）. void push(seqstack *S,int i,datatype x)退栈函数

T(n)=0(1)， S(n)=0(1)；

2）. datatype pop(seqstack *S,int i) 退栈函数

T(n)=0(1)， S(n)=0(1)；

3)．datatype top(seqstack *S,int i)取栈顶函数

T(n)=0(1)， S(n)=0(1)；

4)． void showstack(seqstack *S) 列栈函数

T(n)=0(n)， S(n)=0(1)；

5). main() 主函数

T(n)=O(n) ， S(n)=O(n).

3．改进设想，经验体会

五. 使用说明：如何使用你编制的程序、操作步骤.

编译程序成功后，按界面提示输入进栈数据。

六. 测试结果：

输入输出数据内容：

窗口显示如下：(下划线部分为输入部分，其余为输出部分)

测试数据一：

please push a string for stack0

very↙

please push a string for stack1

happy↙

pop top0 y

pop top1 y

now top0data r

now top1data p

show stack0

r e v

finish stack0

show stack1

p p a h

finish stack1

测试数据二：

please push a string for stack0

↙

please push a string for stack1

↙

stack0underflow

stack1underflow

pop top0

pop top1

stack0empty

stack1empty

now top0data

now top1data

show stack0

finish stack0

show stack1
finish stack1

七．源代码清单

```
#include"stdio.h"
#include"stdlib.h"
#define M 100
typedef char datatype;
typedef struct
{ datatype V[M];int top0,top1;}seqstack;/*两个栈共享向量空间*/

void push(seqstack *S,int i,datatype x)
{if(S->top0==S->top1){printf("overflow");}
else    { if(i==0){S->top0++;S->V[S->top0]=x;}
            if(i==1){S->top1--;S->V[S->top1]=x;}
         }
}

datatype pop(seqstack *S,int i)
{                               if(i==0)
                {if(S->top0<=-1){printf("\nstack0underflow\n");return NULL;}
                else {S->top0--;return S->V[++(S->top0)];}
                }
                                if(i==1)
                {if(S->top1==M){printf("\nstack1underflow\n");return NULL;}
                else {S->top1++;return S->V[--(S->top1)];}
                }
}

datatype top(seqstack *S,int i)
{                               if(i==0)
                {if(S->top0<=-1){printf("\nstack0empty\n");return NULL;}
                else {S->top0--;return S->V[(S->top0)];}
                }
                                if(i==1)
                {if(S->top1==M){printf("\nstack1empty\n");return NULL;}
                else {S->top1++;return S->V[(S->top1)];}
                }
}
void showstack(seqstack *S)
{       int r1=S->top0,r2=S->top1;
        printf("\nshow stack0\n");
while(r1>=0){printf("%c ",S->V[r1]);r1--;}
        printf("\nfinish stack0\n");
```

```
        printf("\nshow stack1\n");
while(r2<M){printf("%c ",S->V[r2]);r2++;}
        printf("\nfinish stack1\n");
}
main()
{int i=0,j=0;char str1[100];
char str2[100];datatype x0,x1;
seqstack *S=(seqstack *)malloc(sizeof(seqstack));/*借空间*/
S->top0=-1;S->top1=M;
printf("please push a string for stack0\n");gets(str1);
printf("please push a string for stack1\n");gets(str2);
while(str1[i]!='\0')
{push(S,0,str1[i]);i++;}
while(str2[j]!='\0')
{push(S,1,str2[j]);j++;}
x0=pop(S,0);x1=pop(S,1);
printf("\npop top0 %c\n",x0);
printf("\npop top1 %c\n",x1);
x0=top(S,0);x1=top(S,1);
printf("\nnow top0data    %c\n",x0);
printf("\nnow top1data    %c\n",x1);
showstack(S);getch();
free(S);/*物归原主*/
}
```