#include<stdio.h>

#include<stdlib.h>

typedef struct Node{

int data;

struct Node \*Next;

}Node,\*LinkList;

typedef struct STACK{

LinkList top;

LinkList Bottom;

}Stack,\*PStack;

void Init(PStack PS);

void push(PStack PS,int val);

void traverse(PStack PS);

int is\_empty(PStack PS);

int pop(PStack PS,int \*pval);

void clear(PStack PS);

int main()

{

Stack S;

int val;

Init(&S);

push(&S,3);

push(&S,88);

push(&S,7);

push(&S,44);

traverse(&S);

if(pop(&S,&val))

printf("出栈成功,您出栈的元素数值为%d\n",val);

else

printf("出栈失败!\n");

traverse(&S);

clear(&S);

traverse(&S);

return 0;

}

//栈初始化

void Init(PStack PS)

{

PS->top = (LinkList)malloc(sizeof(Node));

if(NULL==PS->top)

{

printf("动态分配失败!\n");

exit(-1);

}

else

{

PS->Bottom = PS->top;

PS->top->Next = NULL;

}

return;

}

//入栈

void push(PStack PS,int val)

{

LinkList p = (LinkList)malloc(sizeof(Node));

if(NULL==p)

{

printf("动态分配失败!\n");

exit(-1);

}

p->data = val;

p->Next = PS->top;

PS->top = p;

return;

}

//遍历栈

void traverse(PStack PS)

{

LinkList p = PS->top;

while(p!=PS->Bottom)

{

printf("%d ",p->data);

p = p->Next;

}

printf("\n");

return;

}

int is\_empty(PStack PS)

{

if(PS->top == PS->Bottom)

return 1;

else

return 0;

}

//弹栈

int pop(PStack PS,int \*pval)

{

if(is\_empty(PS))

return 0;

else

{

LinkList p =PS->top;

\*pval = p->data;

PS->top = p->Next;

free(p);

p = NULL;

return 1;

}

}

//清栈

void clear(PStack PS)

{

LinkList p = PS->top;

LinkList q = NULL;

if(is\_empty(PS))

return;

while(NULL!=p)

{

q = p->Next;

free(p);

p = q;

}

PS->top = PS->Bottom;

return;

}