# 顺序存储

## 定义

typedef struct{

int data[MAX\_SIZE];

int top;

}SeqStack;

## 获取元素

int SeqStackGetElement(SeqStack \*s,int \*e){

if(s->top==-1)

return -1;

\*e=s->data[s->top];

return 0;

}

## 插入元素

int SeqStackPush(SeqStack \*s,int \*e){

if(s->top=MAX\_SIZE)

return -1;

s->top++;

s-data[s->top]=\*e;

}

## 删除元素

说明：删除元素需要考虑栈是否为空

int SeqStackPop(SeqStack \*s,int \*e){

if(s->top==0)

return -1;

\*e=s->data[s->top];

s->top--;

return 0;

}

# 链式存储

## 定义

typedef struct LinkStackNode{

int data;

struct LinkStackNode \*next;

}LinkStackNode \*LinkStackPtr;

typedef struct LinkStack{

LinkStackPtr \*top;

int count;

}LinkStack;

## 获取元素

int LinkListGetElement(LinkStack \*s, int \*e){

LinkStackPtr \*p = (LinkStackPtr)malloc(sizeof(LinkStackNode));

p=s->top;

\*e=p->data;

return 0;

}

## 插入元素

int LinkStackPush(LinkStack \*s,int \*e){

LinkStackPtr \*p = (LinkStackPtr)malloc(sizeof(LinkStackNode));

//new一个node

p->data=\*e;

p->next=s->top;

s->top=p;

s->count++;

return 0;

}

## 删除元素

int LinkStackPop(LinkStack \*s,int \*e){

LinkStackPtr \*p;

\*e=s->top->data;

p=s->top;

s->top=p->top->next;

free(s);

s->count--;

return 0;

}