数据结构

（C语言版）（第2版）

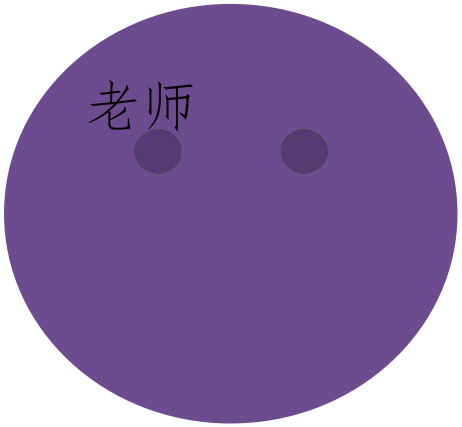
栈和队列

栈的表示和操作

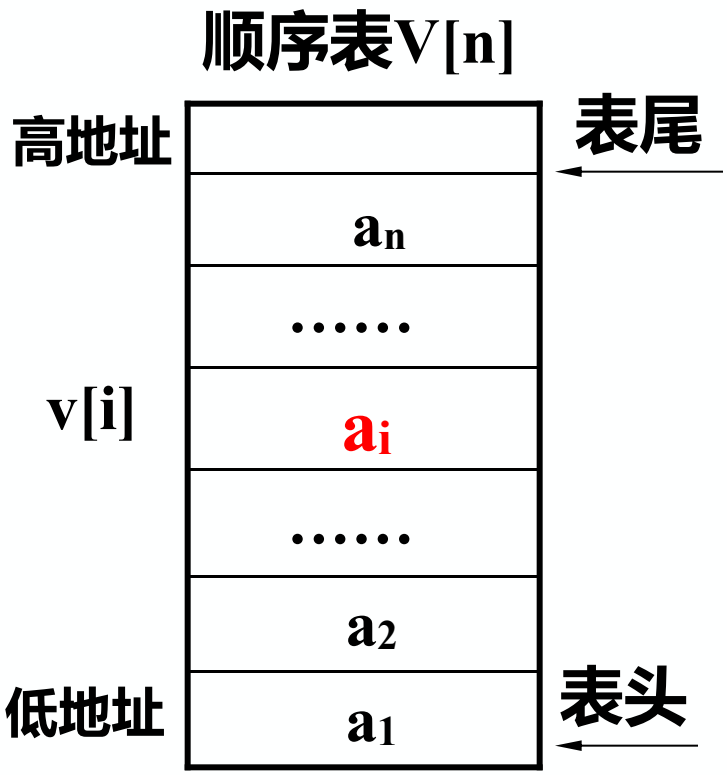
主讲教师：汪红松

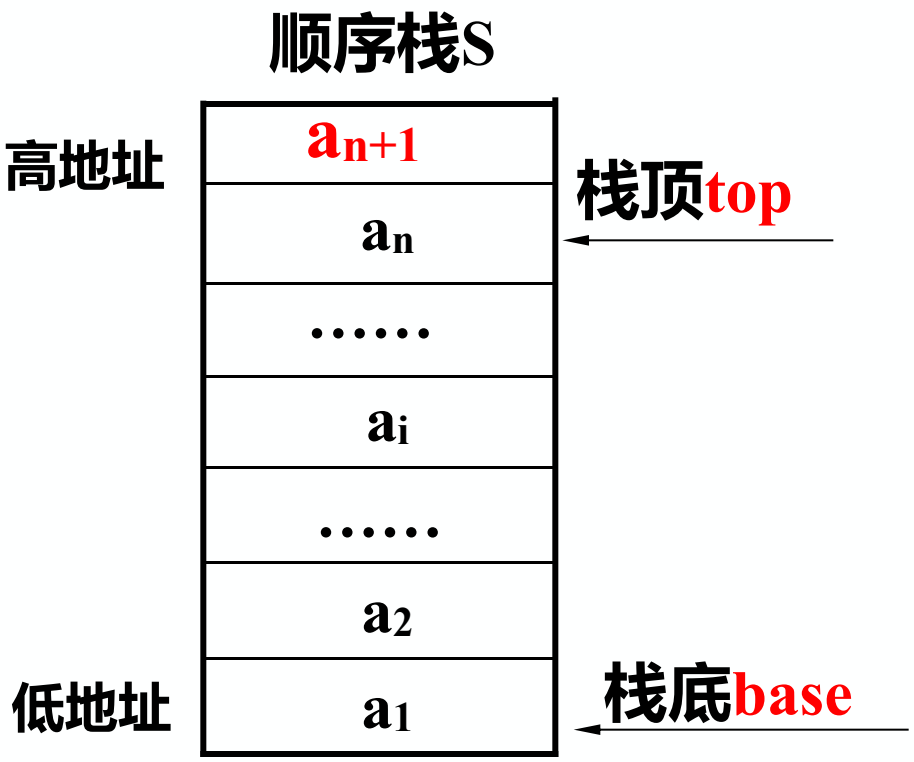
教 学 内 容 Contents

一、顺序栈与顺序表
二、顺序栈
三、链栈



一、顺序栈与顺序表





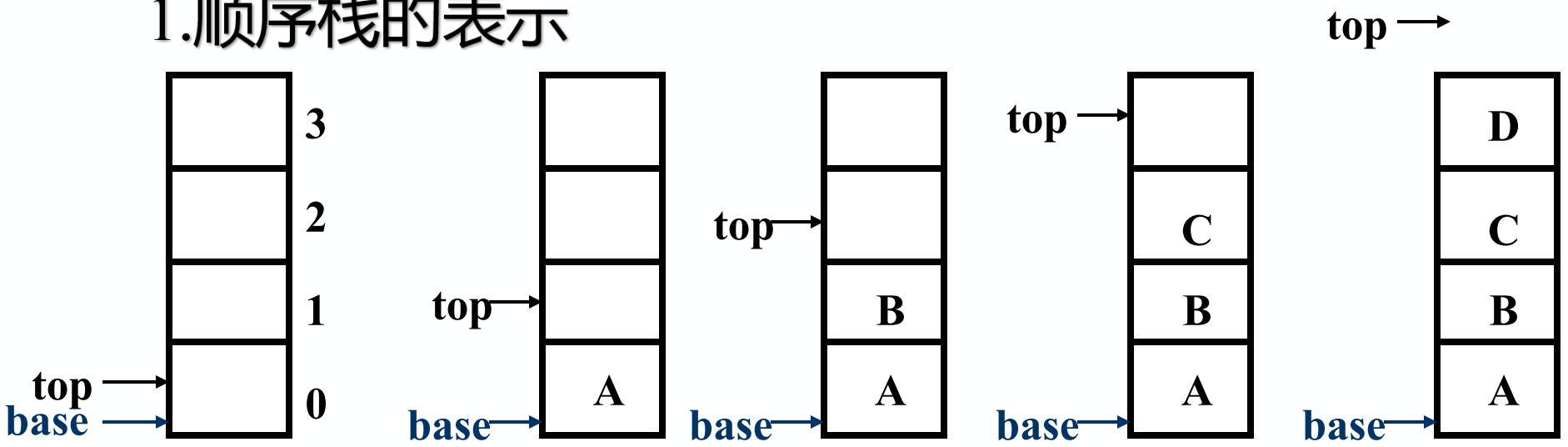
写入：v[i]= a_i
读出： x= v[i]

压入：PUSH (a_{n+1})
弹出： POP (x)

前提：一定要预设栈顶指针top！

二、顺序栈

1.顺序栈的表示



空栈

base == top 是栈空标志

stacksize = 4

top 指示真正的栈顶元素之上的下标地址

栈满时的处理方法：

1、报错,返回操作系统。

2、分配更大的空间，作为栈的存储空间,将原栈的内容移入新栈。

二、顺序栈

1.顺序栈的表示

```
#define  MAXSIZE  100
typedef struct
{
        SElemType   *base;
        SElemType   *top;
        int stacksize;
}SqStack;
```

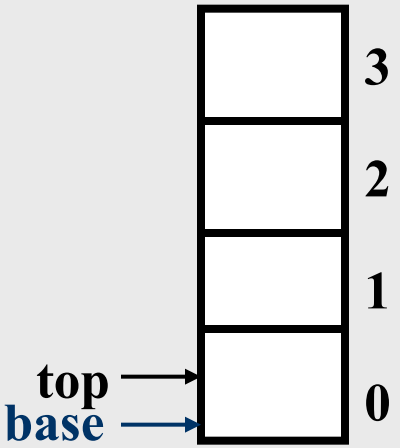
二、顺序栈

2.顺序栈初始化

```
Status InitStack( SqStack &S )
{
    S.base =new SElemType[MAXSIZE] ;
    if( !S.base )        return OVERFLOW;
    S.top = S.base;
    S.stackSize = MAXSIZE;
    return OK;
}
```

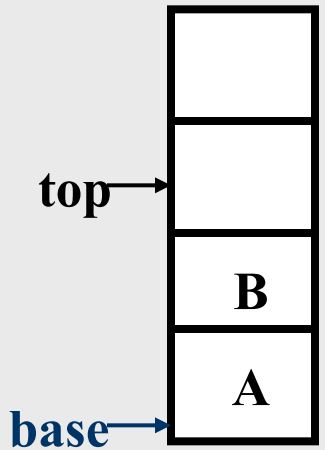
3.判断顺序栈是否为空

```
bool StackEmpty( SqStack S )
{
    if(S.top == S.base) return true;
   else return false;
}
```



4.求顺序栈的长度

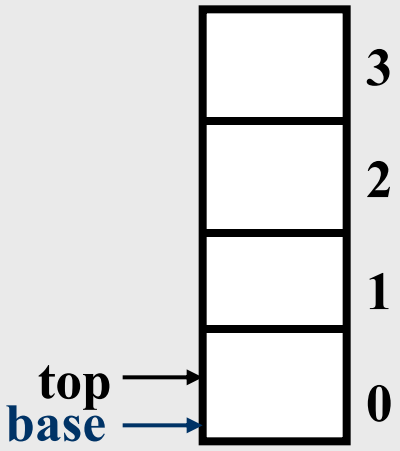
```
int StackLength( SqStack S )
{
    return S.top – S.base;
}
```



二、顺序栈

5.清空顺序栈

```
Status ClearStack( SqStack S )
{
    if( S.base ) S.top = S.base;
    return OK;
}
```



二、顺序栈

6.销毁顺序栈

```
Status DestroyStack( SqStack &S )
{
    if( S.base )
    {
        delete S.base ;
        S.stacksize = 0;
        S.base = S.top = NULL;
    }
  return OK;
}
```

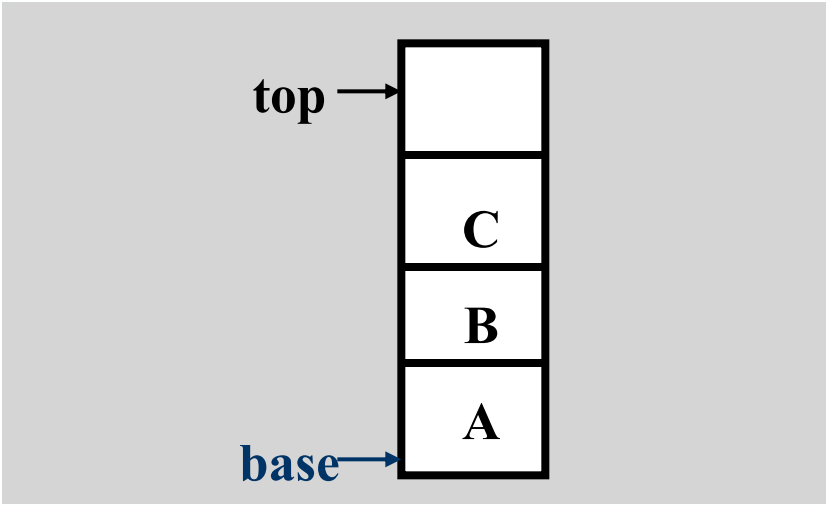
二、顺序栈

7.顺序栈进栈

(1)判断是否栈满，若满则出错；

(2)元素e压入栈顶；

(3)栈顶指针加1。




```
Status Push( SqStack &S, SElemType e)
{
    if( S.top - S.base== S.stacksize )   // ①栈满
       return ERROR;
    *S.top++=e;                          //②③
    return OK;
}
```

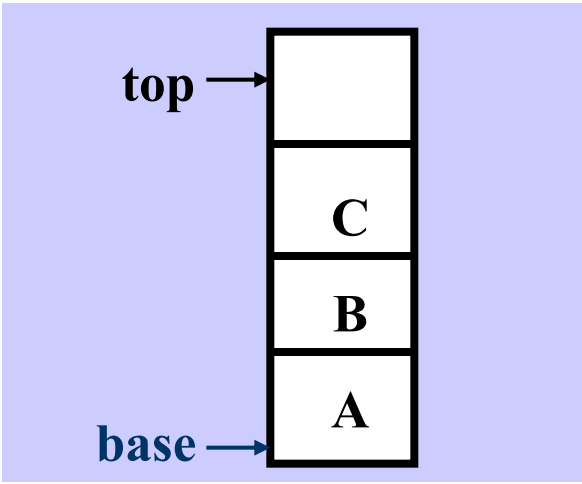
*S.top=e;

S.top++;

二、顺序栈

8.顺序栈出栈

(1)判断是否栈空，若空则出错；
(2)获取栈顶元素e；
(3)栈顶指针减1。




```
Status Pop( SqStack &S, SElemType &e)
{
    if( S.top == S.base ) // ①栈空
        return ERROR;
    e =  *--S.top;          //②③
    return OK;
}
```

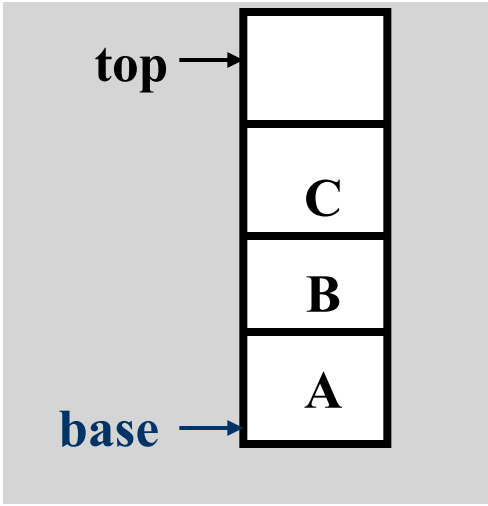
--S.top;
e=*S.top;

二、顺序栈

9.取顺序栈栈顶元素

(1) 判断是否空栈，若空则返回错误;

(2) 否则通过栈顶指针获取栈顶元素。



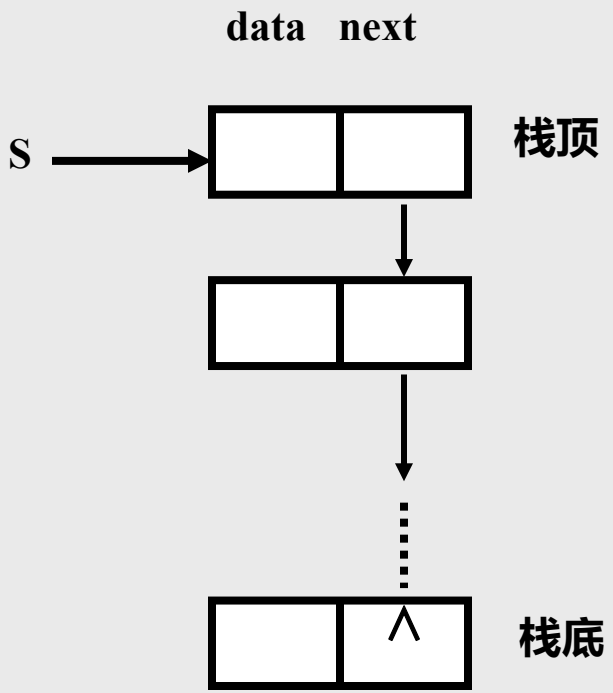

```
Status GetTop( SqStack S, SElemType &e)
{
    if( S.top == S.base )          return ERROR;          // 栈空
    e = *( S.top – 1 );
    return OK;
}
```

三、链栈

1.链栈的表示

- ✓ 运算是受限的单链表，只能在链表头部进行操作，故没有必要附加头结点。栈顶指针就是链表的头指针。

```
typedef  struct StackNode {
      SElemType  data;
      struct StackNode *next;
 } StackNode,  *LinkStack;
LinkStack S;
```



三、链栈

2.链栈的初始化

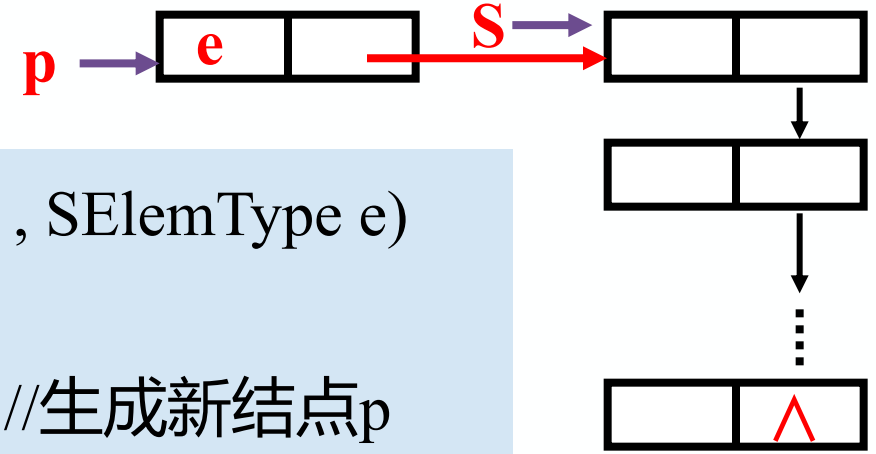

```
void InitStack(LinkStack &S )
{
   S=NULL;
}
```

3.判断链栈是否为空

```
Status StackEmpty(LinkStack S)
  {
    if (S==NULL) return TRUE;
    else return FALSE;
  }
```

三、链栈

4.链栈进栈

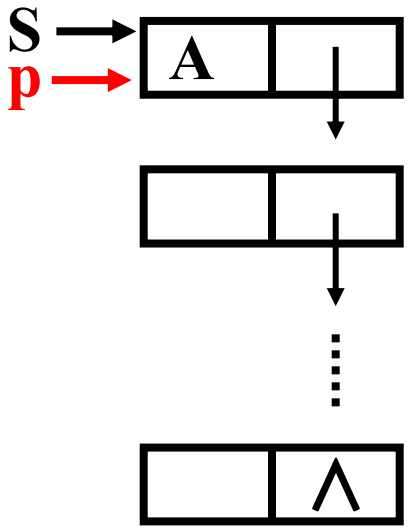



```
Status Push(LinkStack &S , SElemType e)
    {
     p=new StackNode;      //生成新结点p
     if (!p) exit(OVERFLOW);
     p->data=e; p->next=S; S=p;
     return OK;
}
```

三、链栈

5.链栈出栈

e = ‘A’




```
Status Pop (LinkStack &S,SElemType &e)
{
    if (S==NULL) return ERROR;
    e = S-> data;  p = S;   S =  S-> next;
    delete p;   return OK;
}
```

6.取链栈栈顶元素

```
SElemType GetTop(LinkStack S)
  {
     if (S==NULL) exit(1) ;
    else return S–>data;
  }
```

小结

顺序栈和链栈的表示和栈初始化、判栈空栈满、清空与销毁栈、入栈、出栈、取栈顶元素等操作。