栈

一、栈的基本概念

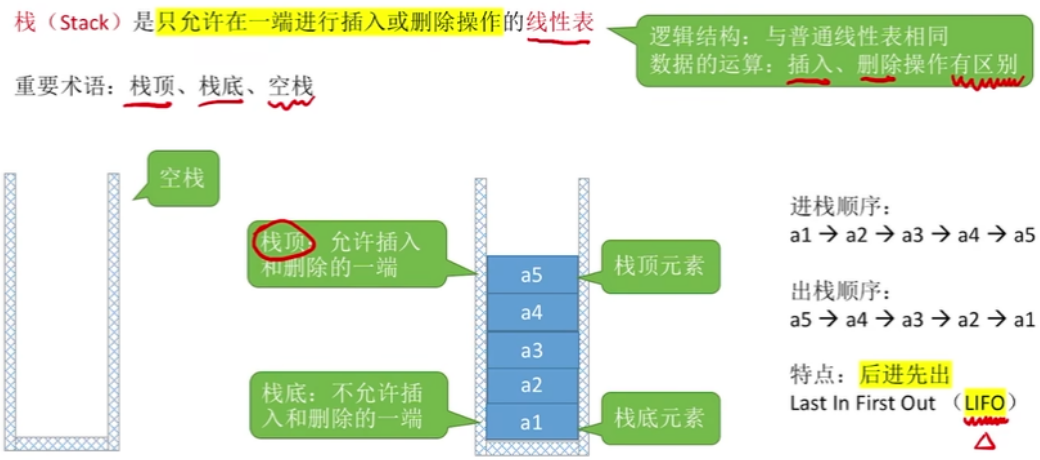
- **栈的定义**

  线性表是具有相同数据类型的n（n>=0）个数据元素的有限序列，其中n为表长，当n=0时线性表示一个空表。若用L命名线性表，则其一般表示为

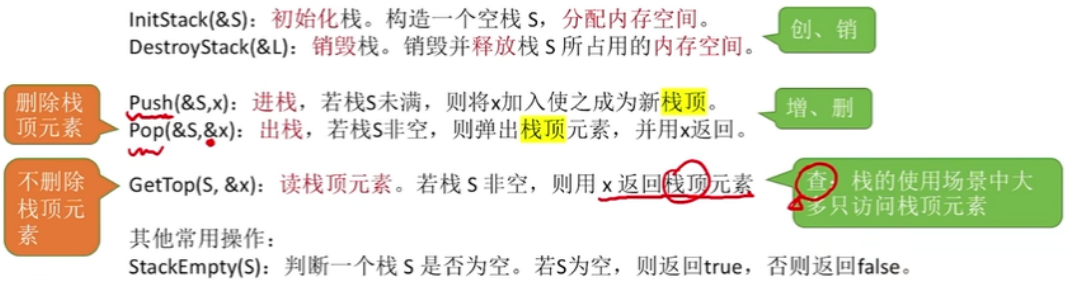
  $$L = (a_1, a_2, \ldots, a_i, a_{i+1}, \ldots, a_n)$$

  栈（Stack）是只允许在一端进行插入或删除操作的线性表

  重要术语：栈顶、栈底、空栈



- **栈的基本操作**



- **栈的常考题型**

进栈顺序：
a→b→c→d→e

有哪些合法的出栈顺序？

e、d、c、b、a
b、e、d、c、a

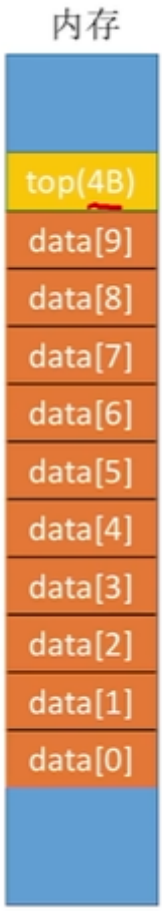
n个不同元素进栈，出栈元素不同排列的个数为$\frac{1}{n+1}C_{2n}^{n}$。
上述公式称为卡特兰（Catalan）数，可采用数学归纳法证明（不要求掌握）。

$$\frac{1}{5+1}C_{10}^{5}=\frac{10*9*8*7*6}{6*5*4*3*2*1}=42$$

二、栈的顺序存储实现

- **顺序栈的定义**

```
#define MaxSize 10            //定义栈中元素的最大个数
typedef struct
{
    ElemType data[MaxSize];  //静态数组存放栈中元素
    int top;                 //栈顶指针
}SqStack;//Sq:sequence

void testStack()
{
    SqStack S;       //声明一个顺序栈（分配空间）
    //...后续操作...
}
```

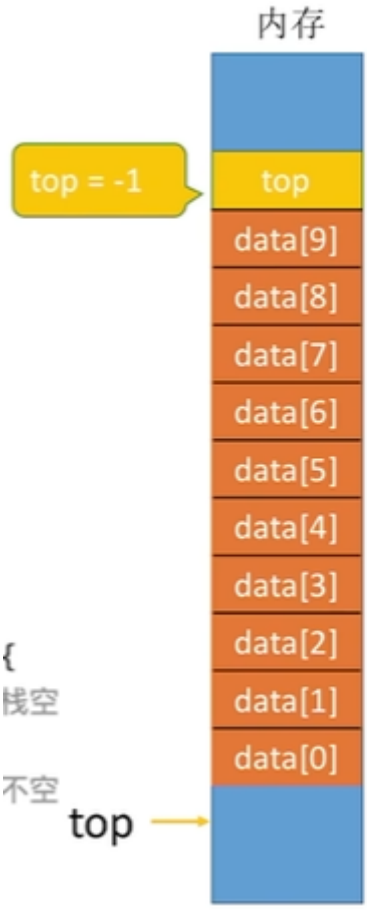


顺序存储：给各个数据元素分配连续的存储空间，大小为MaxSize*sizeof(ElemType)

- **初始化操作**

```
#define MaxSize 10            //定义栈中元素的最大个数
typedef struct
{
    ElemType data[MaxSize];   //静态数组存放栈中元素
    int top;                  //栈顶指针
}SqStack;//Sq:sequence

void InitStack(SqStack &S)    //初始化栈
{
```

```
    S.top = -1;                 //初始化栈顶指针
}

//判断栈空
bool StackEmpty(SqStack S)
{
    if(S.top == -1)     //栈空
        return true;
    else
        return false;   //不空
}

void testStack()
{
    SqStack S;      //声明一个顺序栈（分配空间）
    InitStack(S);
    //...后续操作...
}
```



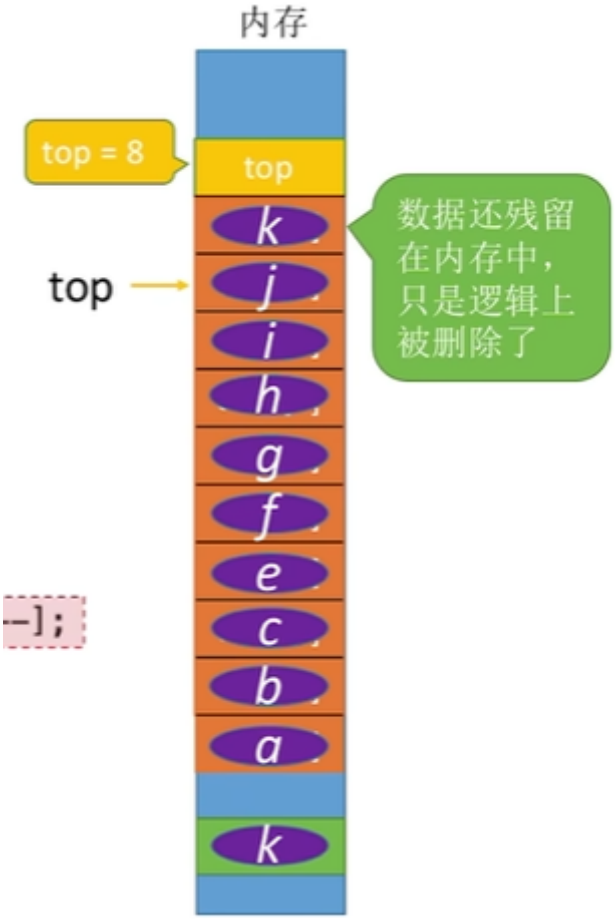
- **进栈操作**

```
#define MaxSize 10              //定义栈中元素的最大个数
typedef struct
{
    ElemType data[MaxSize];  //静态数组存放栈中元素
    int top;                 //栈顶指针
}SqStack;

//新元素入栈
```

```
bool Push(SqStack &S,ElemType x)
{
    if(S.top == MaxSize - 1)  //栈满，报错
        return false;

    S.top = S.top + 1;         //指针先加1
    S.data[S.top] = x;         //新元素入栈
    return true;
}
/*S.top = S.top + 1;
  S.data[S.top] = x;
  等价于
  S.data[++S.top] = x;*/
```

- **出栈操作**

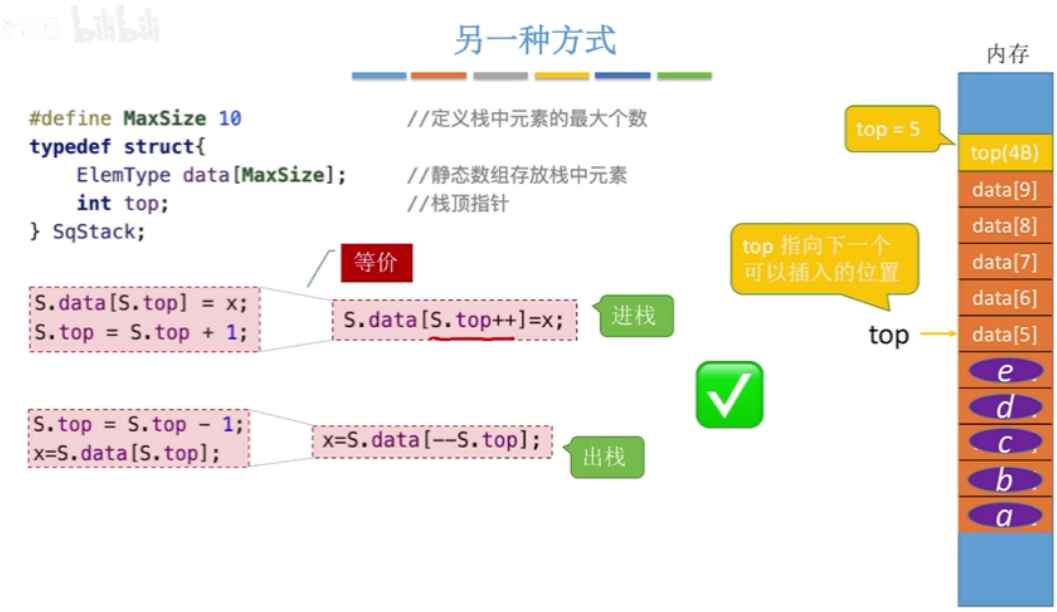
```
#define MaxSize 10            //定义栈中元素的最大个数
typedef struct
{
    ElemType data[MaxSize];  //静态数组存放栈中元素
    int top;                 //栈顶指针
}SqStack;

//出栈操作
bool Pop(SqStack &S,ElemType &x)
{
    if(S.top == -1)        //栈空，报错
        return false;

    x = S.data[S.top];     //栈顶元素先出栈
    S.top = S.top - 1;     //指针再减1
    return true;
}
/*x = S.data[S.top];
  S.top = S.top - 1;
  等价于
  x = S.data[S.top--]*/
```



- **读栈顶元素操作**

```
#define MaxSize 10              //定义栈中元素的最大个数
typedef struct
{
    ElemType data[MaxSize];  //静态数组存放栈中元素
    int top;                 //栈顶指针
}SqStack;

//出栈操作
bool Pop(SqStack &S,ElemType &x)
{
    if(S.top == -1)        //栈空，报错
        return false;

    x = S.data[S.top--];  //先出栈，指针再减1
    return true;
}

//读栈顶元素
bool GetTop(SqStack S,ElemType &x)
{
    if(S.top == -1)        //栈空，报错
        return false;

    x = S.data[S.top];    //x记录栈顶元素
    return true;
}
```

- **另一种方式**

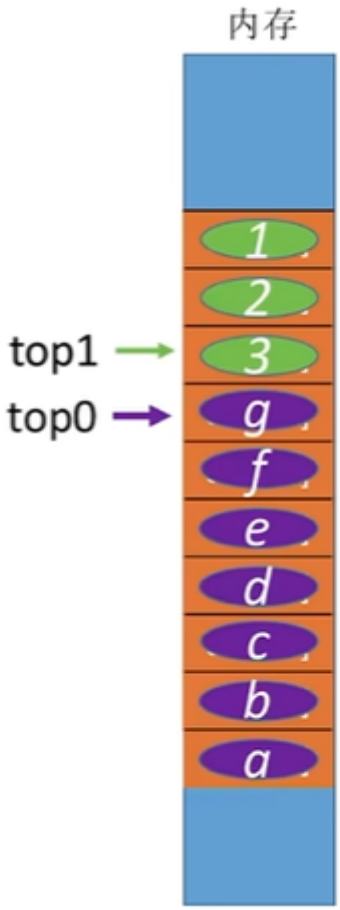


栈满条件：top == MaxSize

顺序栈的缺点：栈的大小不可变

- **共享栈**

```
#define MaxSize 10              //定义栈中元素的最大个数
typedef struct
{
    ElemType data[MaxSize];  //静态数组存放栈中元素
    int top0;                //0号栈顶指针
    int top1;                //1号栈顶指针
}ShStack;

//初始化栈
void InitStack(ShStack &S)
{
    S.top0 = -1;
    S.top1 = MaxSize;
}
```



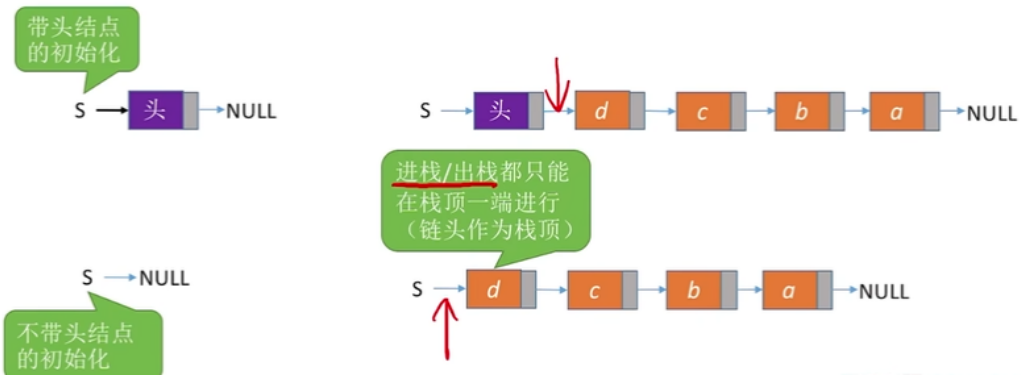
两个栈共享一片空间

栈满条件：top0 + 1 == top1

三、栈的链式存储实现

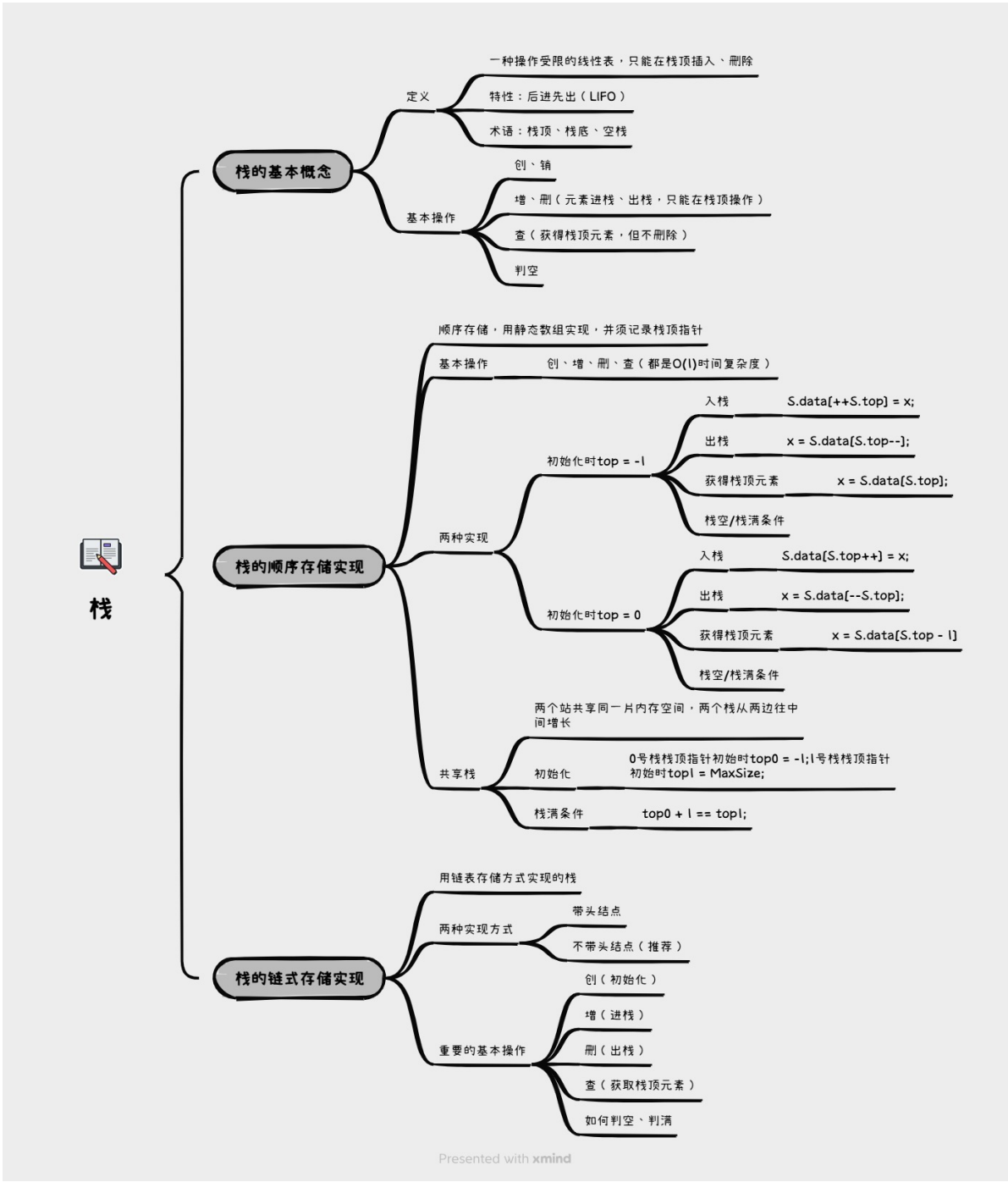
- **链栈的定义**

```
typedef struct Linknode
{
    ElemType data;              //数据域
    struct Linknode *next;      //指针域
}*LiStack;                      //栈类型定义
```



剩下的创、增、删、查、如何判空、判满和单链表类似

四、知识总结


栈
栈的基本概念
定义
一种操作受限的线性表，只能在栈顶插入、删除
特性：后进先出（LIFO）
术语：栈顶、栈底、空栈
基本操作
创、销
增、删（元素进栈、出栈，只能在栈顶操作）
查（获得栈顶元素，但不删除）
判空
栈的顺序存储实现
顺序存储，用静态数组实现，并须记录栈顶指针
基本操作
创、增、删、查（都是O(1)时间复杂度）
两种实现
初始化时top = -1
入栈 S.data[++S.top] = x;
出栈 x = S.data[S.top--];
获得栈顶元素 x = S.data[S.top];
栈空/栈满条件
初始化时top = 0
入栈 S.data[S.top++] = x;
出栈 x = S.data[--S.top];
获得栈顶元素 x = S.data[S.top - 1]
栈空/栈满条件
共享栈
两个站共享同一片内存空间，两个栈从两边往中间增长
初始化
0号栈栈顶指针初始时top0 = -1;1号栈栈顶指针初始时top1 = MaxSize;
栈满条件 top0 + 1 == top1;
栈的链式存储实现
用链表存储方式实现的栈
两种实现方式
带头结点
不带头结点（推荐）
重要的基本操作
创（初始化）
增（进栈）
删（出栈）
查（获取栈顶元素）
如何判空、判满
Presented with xmind