

本节内容

顺序栈的实现

王道考研/CSKAOYAN.COM

1

知识总览

用顺序存储方式实现的栈

顺序栈

基本操作

创、销、增、删、改、查

创（初始化）
增（进栈）
删（出栈）
查（获取栈顶元素）
判空、判满

王道考研/CSKAOYAN.COM

2

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 —— 顺序

```
void testStack() {
```

```
    → SqStack S; //声明一个顺序栈(分配空间)
```

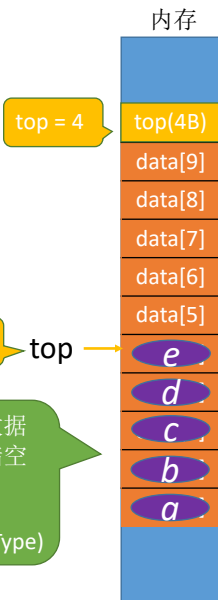
```
    //...后续操作...
```

```
}
```

top 指向栈顶元素

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)



王道考研/CSKAOYAN.COM

3

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;
```

//初始化栈

```
void InitStack(SqStack &S){
```

```
    → S.top = -1;
```

//初始化栈顶指针

```
}
```

```
void testStack() {
```

```
    SqStack S; //声明一个顺序栈(分配空间)
```

```
    → InitStack(S);
```

```
    //...后续操作...
```

```
}
```

增删改查

//判断栈空

```
bool StackEmpty(SqStack S){
```

```
    if(S.top == -1)
```

//栈空

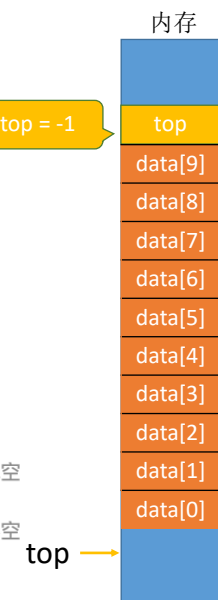
```
        return true;
```

```
    else
```

//不空

```
        return false;
```

```
}
```



王道考研/CSKAOYAN.COM

4

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;
}
```

注意: 错误写法!

`S.data[S.top] = x;`
`S.top = S.top + 1;`

`S.data[++S.top]=x;`

内存

top = 1

top

data[9]

data[8]

data[7]

data[6]

data[5]

data[4]

data[3]

data[2]

b

a

top

王道考研/CSKAOYAN.COM

5

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;
}
```

注意: 错误写法!

`S.data[S.top] = x;`
`S.top = S.top + 1;`

`S.data[++S.top]=x;`

内存

top = 9

top

k

j

i

h

g

f

e

c

b

a

王道考研/CSKAOYAN.COM

6

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;
}
```

等价

$x = S.data[S.top--];$

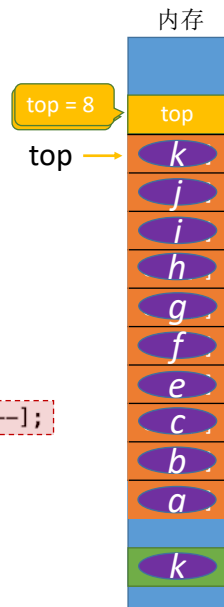
注意: 错误写法!

$S.top = S.top - 1;$
 $x = S.data[S.top];$

$x = S.data[--S.top];$



真的很危险



王道考研/CSKAOYAN.COM

7

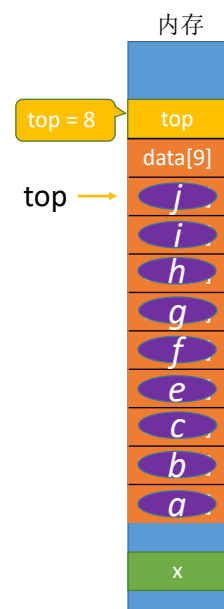
读栈顶元素操作

```
#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;
}
```

唯一区别

```
//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;
}
```



王道考研/CSKAOYAN.COM

8

另一种方式

```
#define MaxSize 10
typedef struct{
    ElemType data[MaxSize];
    int top;
} SqStack;
```

//定义栈中元素的最大个数

//静态数组存放栈中元素
//栈顶指针

//初始化栈

```
void InitStack(SqStack &S){
    S.top=0;
}
```

//初始化栈顶指针

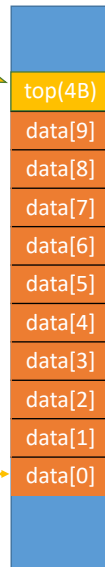
```
void testStack() {
    SqStack S; //声明一个顺序栈(分配空间)
    InitStack(S);
    //...后续操作...
}
```

//判断栈空

```
bool StackEmpty(SqStack S){
    if(S.top==0) //栈空
        return true;
    else //不空
        return false;
}
```

top = 0

内存



空栈

top

王道考研/CSKAOYAN.COM

9

另一种方式

```
#define MaxSize 10
typedef struct{
    ElemType data[MaxSize];
    int top;
} SqStack;
```

//定义栈中元素的最大个数

//静态数组存放栈中元素
//栈顶指针

```
S.data[S.top] = x;
S.top = S.top + 1;
```

等价

```
S.data[S.top++] = x;
```

进栈

```
S.top = S.top - 1;
x = S.data[S.top];
```

```
x = S.data[--S.top];
```

出栈

注意审题 啊喂!

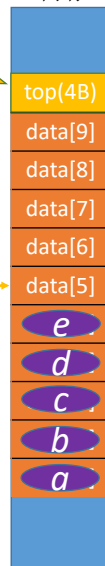
题目不对劲

栈满的条件: top == MaxSize

顺序栈的缺点:
栈的大小不可变

top = 5

内存



top 指向下一个
可以插入的位置

top

不会犯错



王道考研/CSKAOYAN.COM

10

共享栈

两个栈共享一片空间

```
#define MaxSize 10
typedef struct{
    ElemType data[MaxSize];
    int top0;
    int top1;
} ShStack;

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

栈满的条件: $top0 + 1 == top1$

//定义栈中元素的最大个数

//静态数组存放栈中元素

//0号栈栈顶指针

//1号栈栈顶指针

//初始化栈顶指针

内存

top1 →

1

2

3

g

f

e

d

c

b

a

top0 →

王道考研/CSKAOYAN.COM

11

知识回顾与重要考点

顺序栈

- 顺序存储, 用静态数组实现, 并需要记录栈顶指针
- 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/栈满条件是?
 - 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/栈满条件是?
- 共享栈
 - 两个栈共享同一内存空间, 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: $top0 + 1 == top1$;

销? —— 清空、回收

声明栈时分配内存

```
void testStack() {
    SqStack S;
    InitStack(S);
    //...后续操作...
}
```

函数运行结束后系统自动回收内存

不会犯错

王道考研/CSKAOYAN.COM

12