

软件技术基础

第四讲



上讲主要内容

- 链表
- 单链表
- 双链表
- 循环链表



栈

本讲主要内容

栈的定义

顺序栈

链栈

栈的应用

栈

定义



栈是限定仅在表尾进行插入和删除运算的
线性表；

后进先出（**LIFO**）；先进后出（**FILO**）

记作： $S = (a_0, a_1, \dots, a_n)$

栈底元素

栈顶元素

栈

✓ 入栈指插入数据元素。

(a_1, \dots, a_n)	a_{n+1}
---------------------	-----------

$(a_1, \dots, a_n, a_{n+1})$

✓ 出栈指删除数据元素。

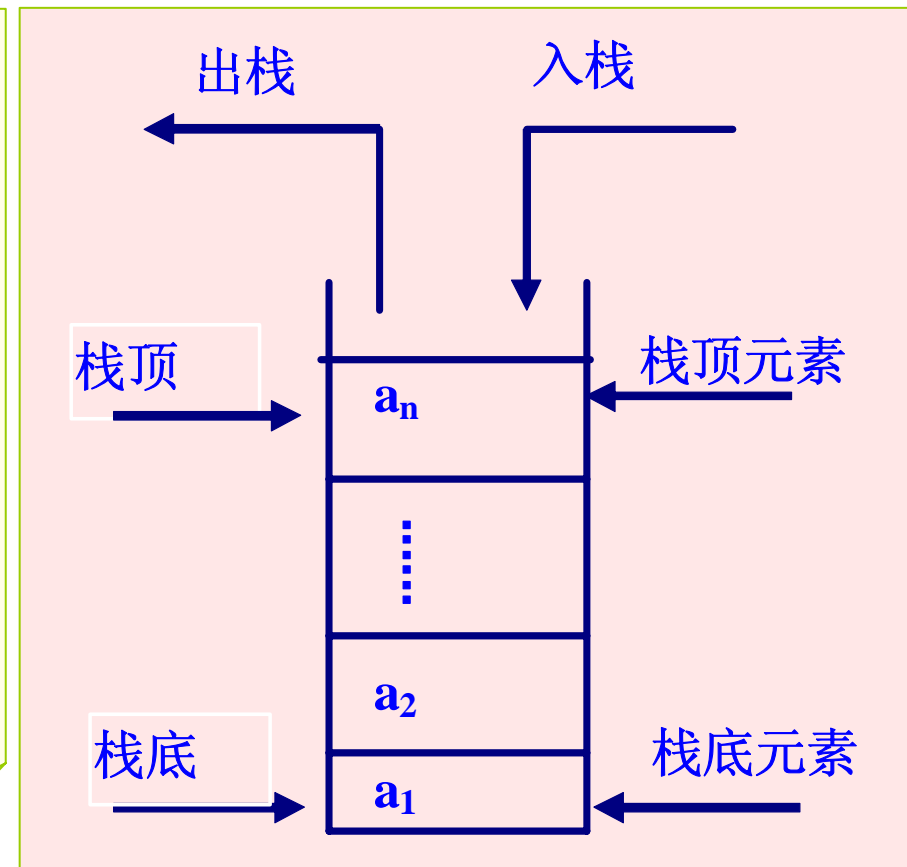
$(a_1, \dots, a_{n-1}, a_n)$

(a_1, \dots, a_{n-1})

栈的基本概念

例 $S=(a_0, a_1, \dots, a_n)$

- 可以形象描述为右图所示形式：
- a_1 是栈底元素；
- a_n 是栈顶元素；
- 入栈指插入数据元素；
- 出栈指删除数据元素；



栈的存储结构

栈的存储结构有两种：

- ❏ 顺序存储结构
采用顺序表存储的栈称为顺序栈。
- ❏ 链式存储结构。
采用单链表存储的栈称为链栈。

顺序栈的定义

栈的顺序存储

```
typedef struct {
```

```
    SElemType *base;
```

```
    SElemType *top;
```

```
    int  stacksize;
```

```
    } SqStack;
```

```
SqStack s;
```

//**base**是栈底指针。

//**top**为栈顶指针

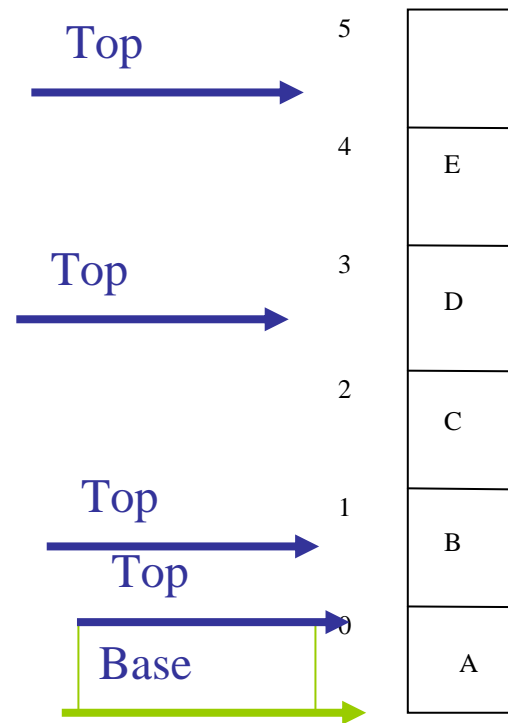
//**stacksize** ——当前栈可使用的最大容量

说明

- base称为栈底指针，始终指向栈底；
当base = NULL时，表明栈结构不存在。
- top为栈顶指针
 - a. top的初始值指向栈底，即**top=base**
 - b. 空栈：当**top=base**时为**栈空**的标记
 - c. 当栈非空时，top的位置：指向当前栈顶元素的**下一个位置**
- stacksize ——当前栈可使用的最大容量

顺序栈操作说明

- (a) 空栈
- (b) A进栈
- (c) BCDE进栈
- (d) ED出栈
- (F) CBA出栈



base==Null 代表栈不存在

说明

- 栈空条件: $s.top = s.base$ 此时不能出栈
- 栈满条件: $s.top - s.base \geq s.stacksize$
- 进栈操作: $*s.top++ = e$; 或 $*s.top = e; s.top++$;
- 退栈操作: $e = *--s.top$; 或 $s.top--; e = *s.top$;
- 当栈满时再做进栈运算必定产生空间溢出,
- 简称“上溢”;
- 当栈空时再做退栈运算也将产生溢出, 简
- 称“下溢”。

栈的初始化

```
Status InitStack (SqStack &S) {  
    S.base = (SElemType )malloc(STACK_INIT_SIZE *  
        sizeof(ElemType));  
    if (!S.base) return (OVERFLOW);  
    S.top=S.base;  
    S.stacksize = STACK_INIT_SIZE;  
    return OK;  
}
```

取栈顶元素

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

进栈

```
Status Push(SqStack &S, SElemType e) {  
    if (S.top-S.base>=S.stacksize)  
    { S.base=(SElemType*)realloc(S.base,  
        (S.stacksize+STACKINCREMENT) *sizeof(ElemType));  
        if (!S.base) return (OVERFLOW);  
        S.top = S.base +S.stacksize;  
        S.stacksize += STACKINCREMENT;  
    }  
    *S.top++ = e;    // *S.top = e; S.top = S.top+1;  
    return OK;  
}
```

出栈

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

链栈的定义

定义

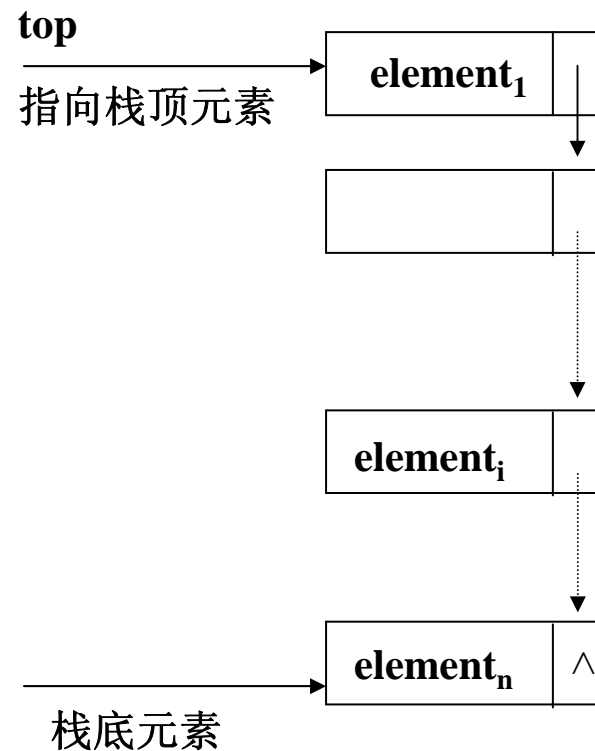
栈的链式存储结构称为**链栈**。它是运算受限的单链表，其**插入**和**删除**操作**仅在表头**进行。

链栈定义如下：

```
typedef struct SNode{  
    SElemType data;  
    struct SNode *next;  
}SNode, *LinkStack;  
LinkStack s;
```

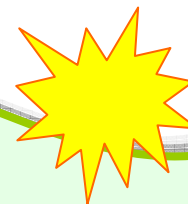

链栈示意图

- ◆ 栈顶是top指针，
它唯一地确定一个链栈。
- ◆ 当top等于NULL时，
该链栈为空栈。



栈的应用

- ✓ 数制转换
- ✓ 行编辑程序
- ✓ 表达式求值




1. 数制转换 p48

十进制N和其它进制数的转换是计算机实现计的基本问题，基于下列原理：

$$N=(n \operatorname{div} d)*d+n \operatorname{mod} d$$

(其中:div为整除运算,mod为求余运算)

例如 $(1348)_{10}=(2504)_8$ ，其运算过程如下：

n	n div 8	n mod 8	
1348	168	4	 低位
168	21	0	
21	2	5	
2	0	2	

高位

算法3.1

要求：输入一个非负十进制整数，输出任意进制数

```
void Conversion()
{ InitStack(s);
  scanf("%d,%d",&N,&base);
  N1=N;
  while (N1)
    { Push(s,N1%base);
      N1 = N1/base; }
  while (!(StackEmpty(s))
    { Pop(s,e);
      if (e>9) printf("%c",e+55);
      else printf("%c",e+48); }
  printf("\n");
}
```

2. 行编辑程序 p50 算法3.2

```
void linedit( )
{  initstack(s);           //初始化堆栈
  ch=getchar( );           //读入一个字符
  while(ch!=eof)           //当不是文章结束符时
  {  while (ch!=eof && ch!='\n')
      {  switch(ch)
          {  case '#' : pop(s,c);
              case '@' : clearstack(s);
              default : push(s,ch);
          }
          ch=getchar( );
      }
      .....;              //完成数据传送
      clearstack(s);        //清空堆栈
      if(ch!=eof)
          ch=gethar( );     //如果文章没有结束，读入下一个字符
  }
  destroystack(s);
}
```

3. 表达式求值 p52 ~ p54

- ✓ 算符间的优先级关系 p52 ~ p53
先括弧内后括弧外
- ✓ 左括号：比括号内的算符的优先级低
比括号外的算符的优先级高
- ✓ 右括号：比括号内的算符的优先级低
比括号外的算符的优先级高
- ✓ #: 优先级总是最低
- ✓ 为实现算符优先算法，可使用两个工作栈：
OPND栈：存数据或运算结果
OPTR栈：存运算符

算法思想：

1. 初态：置OPND栈为空；将“#”作为OPTR栈的栈底元素
2. 依次读入表达式中的每个字符
 - 1) 若是操作数，则进入OPND栈；
 - 2) 若是运算符，则与OPTR栈的栈顶运算符进行优先权（级）的比较：
 - 若读入运算符的优先权高，则进入OPTR栈；
 - 若读入运算符的优先权低，则OPTR退栈（退出原有的栈顶元素），OPND栈退出两个元素（先退出b，再退出a），中间结果再进入OPND栈；
 - 若读入“）”，OPTR栈的原有栈的栈顶元素若为“（”，则OPTR退出“（”；
 - 若读入“#”，OPTR栈栈顶元素也为“#”，则OPTR栈退出“#”，结束。

递归调用

定义：一个过程通过调用语句直接或间接调用自身的过程

一次调用需要保存的信息构成一个工作记录：

- ✕ 返回地址：上层本次调用语句的后接处
- ✕ 本次调用时，与形参结合的实参值
- ✕ 本层的局部变量

递归调用

Fibonacci序列定义为：

$$\text{Fib}(n) = \begin{cases} 0 & (n=0) \\ 1 & (n=1) \\ \text{Fib}(n-1) + \text{Fib}(n-2) & (n>1) \end{cases}$$

小结

顺序栈

链栈

栈的应用

数制转换

行编辑程序

表达式求值

作业

P24

18; 24; 28; 30