西安电子科技大学
XIDIAN UNIVERSITY

第三章
堆栈

STACK

内容大纲

- 3-1 栈的定义与运算
- 3-2 栈的存储和实现
- 3-3 栈的应用举例
- 小 结
- 实验3 栈子系统
- 习题3

数据结构

- 集合结构
- 线性结构 → 线性表、堆栈
- 树形结构
- 图形结构

表达式求值的问题

- 【例】算术表达式5+6/2-3*4。正确理解：

5+6/2-3*4 = 5+3-3*4 = 8-3*4 = 8-12 = -4

- 由两类对象构成：
  - 运算数，如2、3、4
  - 运算符号，如+、-、*、/
- 不同运算符号优先级不一样，求值运算带来难度

如何解决？

后缀表达式

- 中缀表达式：运算符号位于两个运算数之间。如 ，
  a+b*c-d/e
- 后缀表达式：运算符号位于两个运算数之后。如，
  abc*+de/-

填空题 1分

还有一种表达式叫“前缀表达式”，即运算符号位于运算数之前,比如a+b*c的前缀表达式是+a*bc。

你能写出a+b*c-d/e的前缀表达式吗?

[填空1]

6

后缀表达式

- 中缀表达式：运算符号位于两个运算数之间。
- 后缀表达式：运算符号位于两个运算数之后。

【例】62/3-42*+=

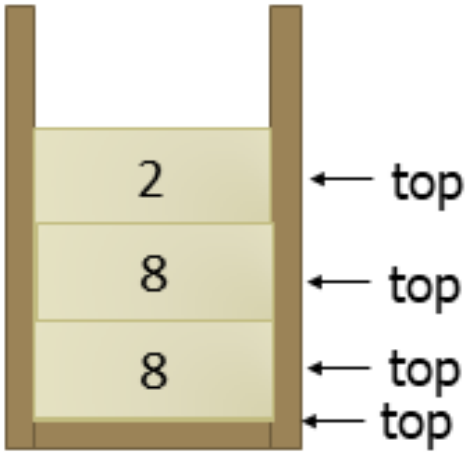
后缀表达式求值策略：从左向右“扫描”，逐个处理运算数和运算符号

1. 遇到运算数怎么办？如何“记住”目前还不未参与运算的数？
2. 遇到运算符号怎么办？对应的运算数是什么？

启示： 需要有种存储方法，能顺序存储运算数，并在需要时“倒序”输出！

后缀表达式

【例】6 2 / 3 – 4 2 * + =



对象: 6 (运算数)	对象: 2 (运算数)
对象: / (运算符)	对象: 3 (运算数)
对象: – (运算符)	对象: 4 (运算数)
对象: 2 (运算数)	对象: * (运算符)
对象: + (运算符)	Pop: 8

T(n)=O(n)

什么是堆栈

- 堆栈（Stack）：具有一定操作约束的线性表
  - 只在一端（栈顶，Top）做 **插入、删除**

- 插入数据：**入栈（push）**
- 删除数据：**出栈（pop）**
- 后入先出：**Last In First Out（LIFO）**

- 应用实例？

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY 9

堆栈的抽象数据类型描述

类型名称: 堆栈（Stack）

数据对象集：一个有0个或多个元素的有穷线性表。

操作集：长度为MaxSize的堆栈S ∈ Stack，堆栈元素item ∈ ElementType

1、Stack CreateStack(int MaxSize)： 生成空堆栈，其最大长度为MaxSize；

2、int IsFull(Stack S, int MaxSize)：判断堆栈S是否已满；

3、void Push(Stack S, ElementType item)：将元素item压入堆栈；

4、int IsEmpty (Stack S)：判断堆栈S是否为空；

5、ElementType Pop(Stack S)：删除并返回栈顶元素；

堆栈的基本运算

1. 进栈：Push（&s，x）

初始条件：栈s已存在且非满。

操作结果：在栈顶插入一个元素x，栈中多了一个元素。

2. 出栈：Pop（&s）

初始条件：栈s存在且非空。

操作结果：删除栈顶元素，栈中少了一个元素。

3. 读栈顶元素：ReadTop（s，&e）

初始条件：栈s已存在且非空。

操作结果：输出栈顶元素，但栈中元素不变。

堆栈的基本运算

4. 判栈空：SEmpty（s）

初始条件：栈s已存在。

操作结果：若栈空则返回为0，否则返回为1。

5. 判栈满：SFull（s）

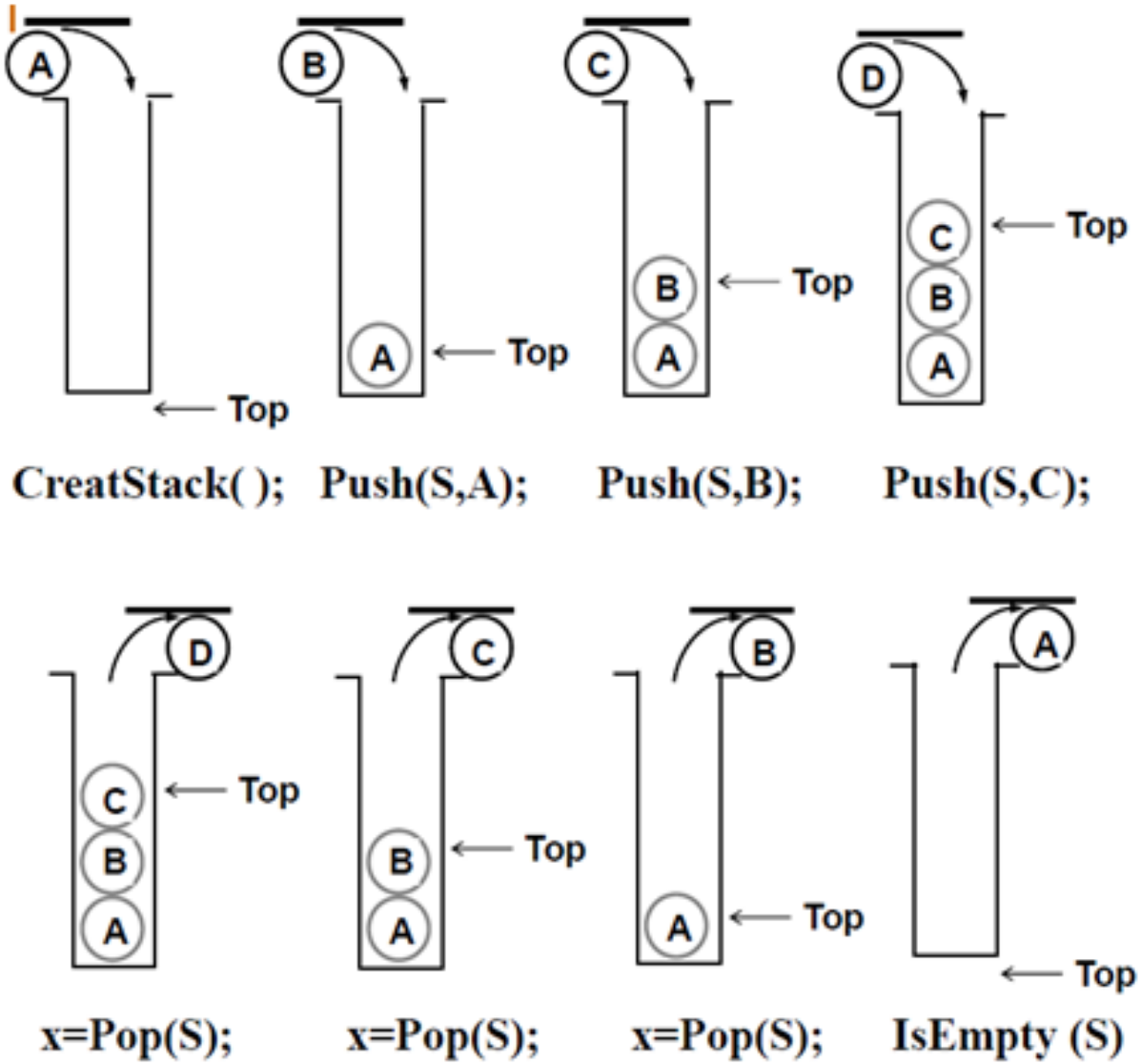
初始条件：栈s已存在。

操作结果：若栈满则返回为0，否则返回为1。

6. 显示栈元素：ShowStack(s)

初始条件：栈s已存在，且非空。

操作结果：显示栈中所有元素。


A
Top
CreatStack( );
B
A
Top
Push(S,A);
C
B
A
Top
Top
Push(S,B);
D
C
B
A
Top
Top
Push(S,C);
D
C
B
A
Top
Top
x=Pop(S);
C
B
A
Top
Top
x=Pop(S);
B
A
Top
Top
x=Pop(S);
A
Top
IsEmpty (S)

- Push和Pop可以穿插交替进行
- 按照操作系列

(1) Push(S,A), Push(S,B), Push(S,C), Pop(S), Pop(S), Pop(S)堆栈输出是?

CBA

(2) Push(S,A), Pop(S), Push(S,B), Push(S,C), Pop(S), Pop(S)堆栈输出是?

ACB

【例】如果三个字符按ABC顺序压入堆栈，ABC的所有排列都可能是出栈的序列吗？

单选题 1分

按ABC顺序入栈，可以产生CAB这样的出栈序列

A 可以

B 不行

15

堆栈的顺序存储实现

栈的顺序存储结构通常由一个一维数组和一个记录栈顶元素位置的变量组成。

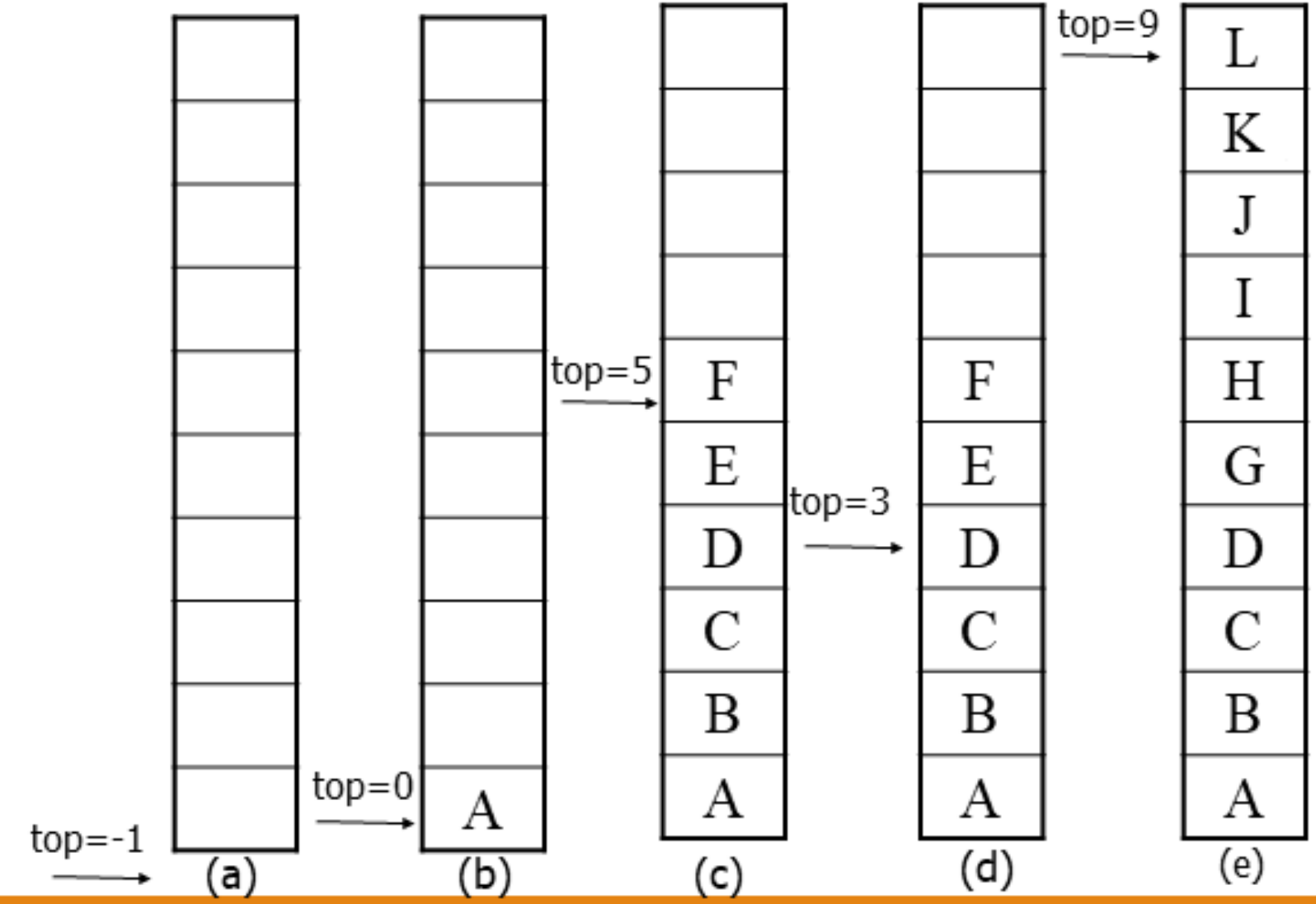
(1) 用一维数组实现

```
#define  MAXLEN  <最大个数>    // 分配最大的栈空间
 char  s[MAXLEN];              // 数据类型为字符型
 int  top;                     // 定义栈顶指针
```

(2) 用结构体数组实现

```
#define MaxSize  <储存数据元素的最大个数>
typedef  struct SNode *Stack;
struct SNode{
        ElementType Data[MaxSize];
        int Top;
};
```

（3）栈操作的示意图



GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY

顺序栈运算的基本算法

（1）置空栈

首先建立栈空间，然后初始化栈顶指针。

```
SeqStack *Snull（）
{ SeqStack *s;
  s=malloc(sizeof(SeqStack)); //分配数组空间
  s->top=-1;                  //置栈空
  return s;
}
```

顺序栈运算的基本算法

（2）进栈

进栈运算是在栈顶位置插入一个新元素x，其算法步骤为：

(a) 判栈是否为满，若栈满，作溢出处理，并返回0；

(b) 若栈未满，栈顶指针top加1；

(c) 将新元素x送入栈顶，并返回1。

```
int Push (SeqStack *s, datatype x)
 {if （s->top==MAXLEN-1）return 0;
                                  // 栈满不能入栈，且返回0
   else { s->top++;
          s->data[s->top]=x;  // 栈不满，则压入元素x
          return 1; }          // 进栈成功，返回1
 }
```

顺序栈运算的基本算法

（3）出栈

出栈运算是指取出栈顶元素，赋给某一个指定变量x，其算法步骤为：

(a) 判栈是否为空，若栈空，作下溢处理，并返回0；

(b) 若栈非空，将栈顶元素赋给变量x；

(c) 指针top减1，并返回1。

```
int Pop（SeqStack *s， datatype *x）
 { if (SEmpty ( s )) return 0;  // 若栈空不能出栈，且返回0
   else { *x=s->data[s->top];   // 栈不空则栈顶元素存入*x
          s->top - -;
          return 1; }           // 出栈成功，返回1
 }
```

顺序栈运算的基本算法

（4）读栈顶元素

```
datatype ReadTop（SeqStack *s）
{ if (SEmpty(s)) return 0;          //若栈空，则返回0
  else return (s->data[s->top]);
                                    //否则，读栈顶元素，但指针未移动
}
```

（5）判栈空

```
int SEmpty（SeqStack *s）
{ if （s->top==–1） return 1;       //若栈空，则返回1
   else return 0;                   //否则返回0
}
```

（6）判栈满

```
int SFull（SeqStack *s）
 { if （s->top==MAXLEN–1） return 1; //若栈满，则返回1
   else return 0;                    //否则返回0
 }
```

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY

主观题 10分

请用一个数组实现两个堆栈，要求最大地利用数组空间，使数组只要有空间入栈操作就可以成功。

```
#define MaxSize <存储数据元素的最大个数>
struct DStack {
    ElementType Data[MaxSize];
    int Top1;       /* 堆栈 1 的栈顶指针 */
    int Top2;       /* 堆栈 2 的栈顶指针 */
} S;

S.Top1 = -1;
S.Top2 = MaxSize;
```

22

```
void Push( struct DStack *PtrS, ElementType item, int Tag )
{  /* Tag作为区分两个堆栈的标志，取值为1和2 */
   if ( PtrS->Top2 - PtrS->Top1 == 1) {   /*堆栈满*/
    printf("堆栈满");    return ;
   }
   if ( Tag == 1 ) /* 对第一个堆栈操作 */
      PtrS->Data[++(PtrS->Top1)] = item;
   else                    /* 对第二个堆栈操作 */
      PtrS->Data[--(PtrS->Top2)] = item;
}
```

```
ElementType Pop( struct DStack *PtrS, int Tag )
{  /* Tag作为区分两个堆栈的标志，取值为1和2   */
    if ( Tag == 1 ) {  /* 对第一个堆栈操作   */
      if ( PtrS->Top1 == -1 ) { /*堆栈1空 */
         printf("堆栈1空");    return NULL;
      } else return PtrS->Data[(PtrS->Top1)--];
   } else {                /* 对第二个堆栈操作 */
      if ( PtrS->Top2 == MaxSize ) { /*堆栈2空 */
         printf("堆栈2空");  return NULL;
      } else  return PtrS->Data[(PtrS->Top2)++];
  }
}
```

堆栈的链式存储实现

栈的链式存储结构实际上就是一个单链表，叫做链栈。
插入和删除操作只能在链栈的栈顶进行。

栈顶指针Top应该在链表的哪一头？

由于栈中的操作只能在栈顶进行的，所以用链表的头部做栈顶是最合适的。


```
typedef struct stacknode *Linkstack;
                          //Linkstack为struct stacknode * 类型的别名
struct stacknode{
   datatype data;          //定义数据类型
   struct stacknode *next;  //定义一个结构体的链指针
};
Linkstack top;             //top为栈顶指针
```

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY

链栈运算的基本算法

(1) 建立空栈（初始化）

```
Stack CreateStack()
{   /* 构建一个堆栈的头结点，返回指针 */
   Stack S;
   S =(Stack)malloc(sizeof(struct SNode));
   S->Next = NULL;
   return S;
}
```

(2) 判断堆栈s是否为空

```
int IsEmpty(Stack S)
{   /*判断堆栈S是否为空，若为空函数返回整数1，否
则返回0 */
    return ( S->Next == NULL );
}
```

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY

链栈运算的基本算法

(3) 入栈

```
void Push(linkstack *s,int x)
  {     stacknode *p=new stacknode; // 申请一个新结点
        p->data=x;                  // 数据入栈
        p->next=s->next;                  // 修改栈顶指针
        s->next=p;
   }
```

链栈运算的基本算法

(4) 出栈

```
int Pop(linkstack *s)
 { int x;                          // 定义一个变量x，用以存放出栈的元素
   stacknode *p=s->next;           // 栈顶指针指向p
   x=p->data;                      // 栈顶元素送x
   s->next=p->next;                // 修改栈顶指针
   delete p;                       // 回收结点p
   return x;                       // 返回栈顶元素
 }
```

链栈运算的基本算法

(5) 显示

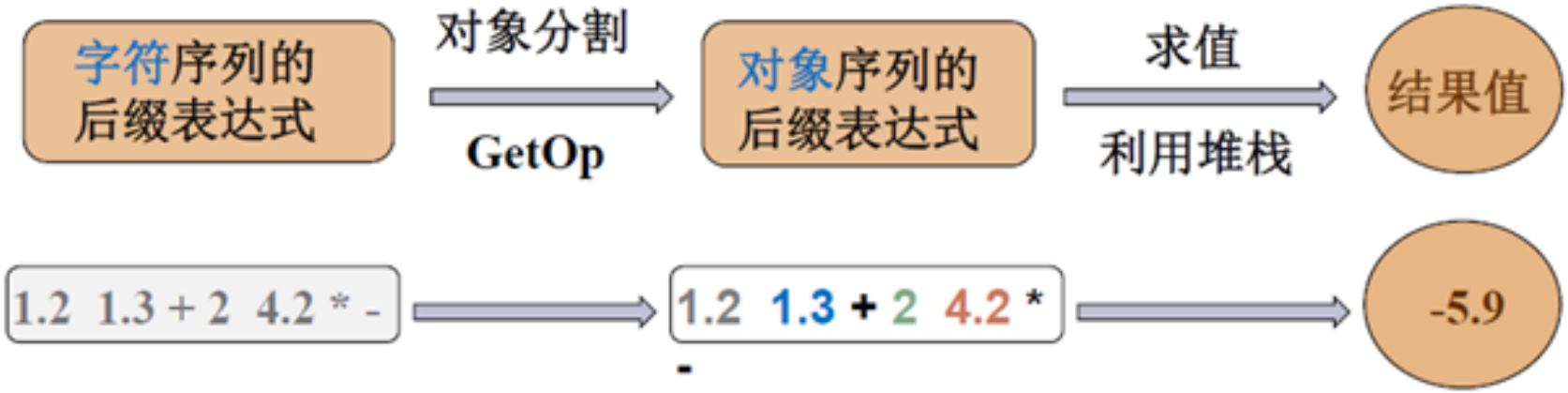
```
void ShowStack (linkstack *s)
 {  stacknode *p=s->next;
    if (p= =NULL)                    // 若栈空，作"栈空"显示
       printf ("栈为空");
    else
      { printf ("栈元素为: ");
        while (p!=NULL)              //栈非空，则显示栈中所有元素
      {   printf ("%6d",p->data); //输出一个元素
          p=p->next; }               // 修改栈顶指针
          printf ("\n");
      }
   }
```

链栈应用：表达式求值

- 回忆：应用堆栈实现后缀表达式求值的基本过程

从左到右读入后缀表达式的各项（运算符或运算数）

- 运算数：入栈
- 运算符号：从堆栈中弹出适当数量的运算数，计算并结果入栈；
- 最后，堆栈顶上的元素就是表达式的结果值



中缀表达式求值

基本策略：将中缀表达式转换为后缀表达式，然后求值

如何将中缀表达式转换为后缀？

【例】2+9/3-5 → 2 9 3 / + 5 –

1. 运算数相对顺序不变
2. 运算符号顺序发生改变
   - 需要存储“等待中”的运算符号
   - 要将当前运算符号与“等待中”的最后一个运算符号比较

堆栈！

输出：2 9 3 / + 5 -

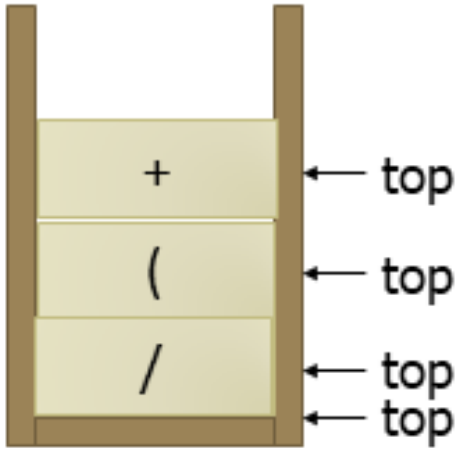
记下：+ /

有括号怎么办？

中缀表达式求值

【例】a*(b+c)/d = ? → a b c + * d /

输出: a b c + * d /



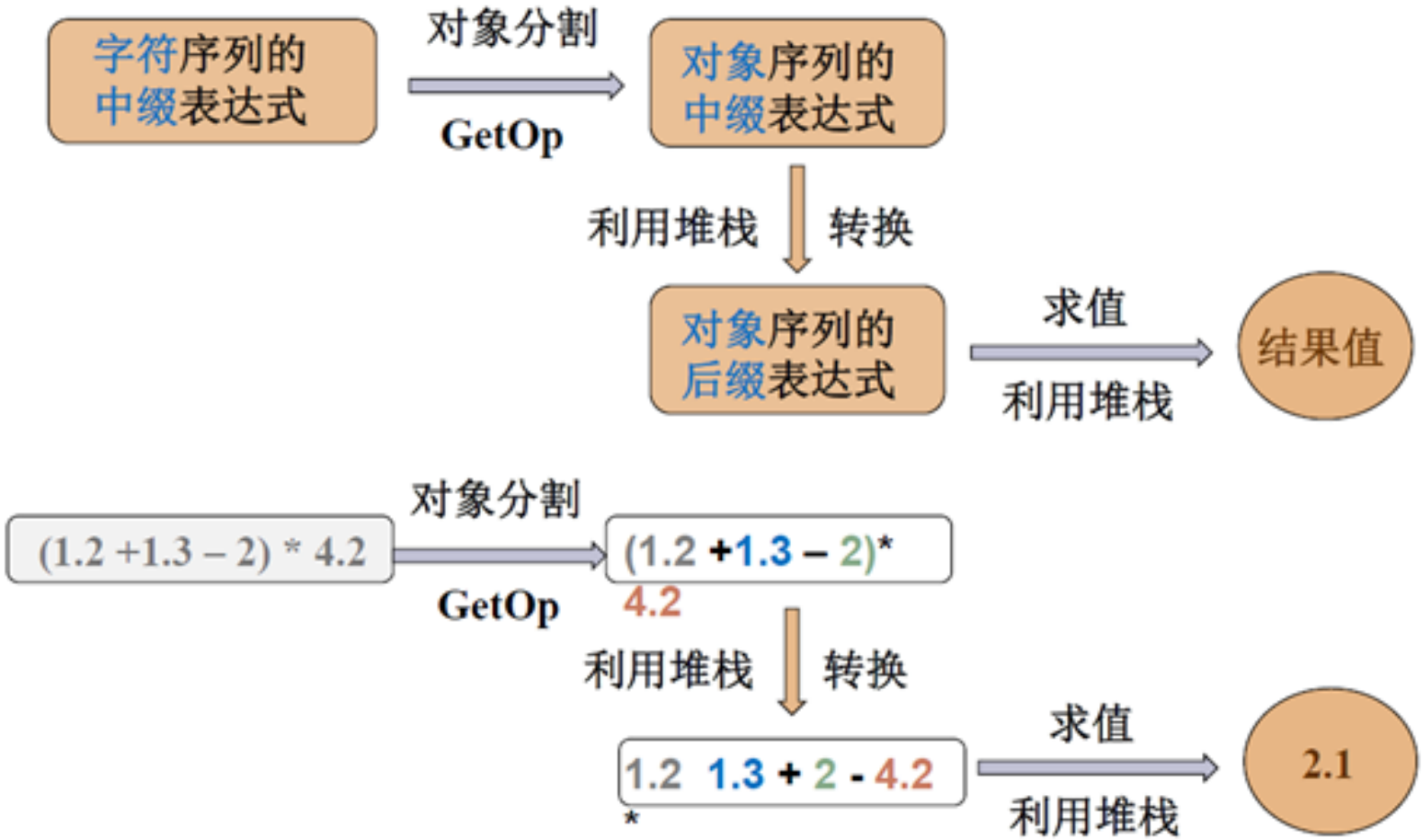
输入对象: *a* (操作数)	输入对象: * (乘法)
输入对象: ((左括号)	输入对象: *b* (操作数)
输入对象: + (加法)	输入对象: *c* (操作数)
输入对象:) (右括号)	输入对象: / (除法)
输入对象: *d* (操作数)	

No!

T(n)=O(n)

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY 31

中缀表达式求值



中缀表达式转换为后缀

- 从头到尾读取中缀表达式的每个对象，对不同对象按不同的情况处理。
  - 运算数：直接输出；
  - 左括号：压入堆栈；
  - 右括号：将栈顶的运算符弹出并输出，直到遇到左括号（出栈，不输出）；
  - 运算符号：
    - 若优先级大于栈顶运算符时，则把它压栈；
    - 若优先级小于等于栈顶运算符时，将栈顶运算符弹出并输出；再比较新的栈顶运算符，直到该运算符大于栈顶运算符优先级为止，然后将该运算符压栈；
  - 若各对象处理完毕，则把堆栈中存留的运算符一并输出。

中缀表达式转换为后缀示例:【例1】 (2*(9+6/3-5)+4)

步骤	待处理表达式	堆栈状态 (底←→顶)	输出状态
1	2*（9+6/3-5）+4		
2	*（9+6/3-5）+4		2
3	（9+6/3-5）+4	*	2
4	9+6/3-5）+4	* (	2
5	+6/3-5）+4	* (	2 9
6	6/3-5）+4	* (+	2 9
7	/3-5）+4	* (+	2 9 6
8	3-5）+4	* (+ /	2 9 6
9	-5）+4	* (+ /	2 9 6 3
10	5）+4	* (-	2 9 6 3 / +
11	）+4	* (-	2 9 6 3 / + 5
12	+4	*	2 9 6 3 / + 5 -
13	4	+	2 9 6 3 / + 5 - *
14		+	2 9 6 3 / + 5 - * 4
15			2 9 6 3 / + 5 - * 4 +

【例2】A / B ^ C + D * E–A*C

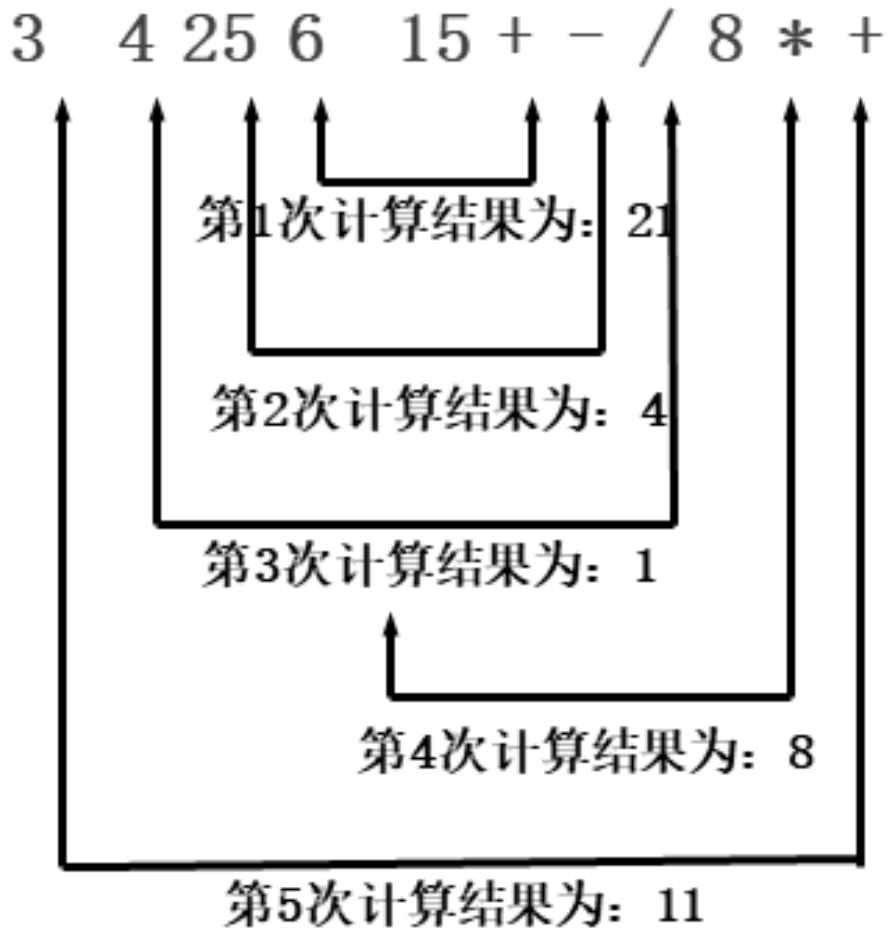
输入符号	运算符栈	输出结果	操作说明
A		A	输出A
/	/	A	/ 进栈
B	/	A,B	输出B
^	/, ^	A,B	^优先级高于/，继续进栈
C	/, ^	A,B,C	输出C
+	+	A,B,C, ^,/	^，/依次弹出
D	+	A,B,C, ^,/,D	输出D
*	+,*	A,B,C, ^,/,D	*优先级高于+，继续进栈
E	+,*	A,B,C, ^,/,D,E	输出E
-	-	A,B,C, ^,/,D,E,*,+	*，+依次弹出，-进栈
A	-	A,B,C, ^,/,D,E,*,+,A	输出A
*	-,*	A,B,C, ^,/,D,E,*,+,A	*优先级高于 -，继续进栈
C	-,*	A,B,C, ^,/,D,E,*,+,A,C	输出C
#		A,B,C, ^,/,D,E,*,+,A,C,*, -	遇到结束符#，依次弹出*，-

【例3】3 + 4 / （25–（6 + 15））* 8

输入符号	运算符栈	输出结果	操作说明
3		3	输出3
+	+	3	+进栈
4	+	3,4	输出4
/	+,/	3,4	/继续进栈
(	+,/,(	3,4	（进栈
25	+,/,(	3,4,25	输出25
-	+,/,(,-	3,4,25	-进栈
(	+,/,(,-,(	3,4,25	（再进栈
6	+,/,(,-,(	3,4,25,6	输出6
+	+,/,(,-,(,+	3,4,25,6	+进栈
15	+,/,(,-,(,+	3,4,25,6,15	输出15
)	+,/,(,-	3,4,25,6,15,+	遇），依次弹出第2个（后的符号
)	+,/	3,4,25,6,15,+,-	遇），依次弹出第1个（后的符号
*	+,*	3,4,25,6,15,+,-,/	弹出/，但*高于+，继续进栈
8	+.*	3,4,25,6,15,+,-,/,8	输出8
#		3,4,25,6,15,+,-,/,8,*,+	遇到结束符#，依次弹出*，+

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY

后缀表达式求值的计算过程：



链栈应用：数制转换

数值进位制的换算是计算机实现计算和处理的基本问题。比如将十进制数N转换为j进制的数，其解决的方法很多，其中一个常用的算法是除j取余法。将十进制数每次除以j，所得的余数依次入栈，然后按“后进先出”的次序出栈便得到转换的结果。

其算法原理是：

$$N = (N / j) * j + N \% j$$

其中：/为整除，%为求余

【例】将十进制数138转换为二进制数

N	N/2（整除）	N%2（求余）
138	69	0
69	34	1
34	17	0
17	8	1
8	4	0
4	2	0
2	1	0
1	0	1

进栈次序 ↑　　出栈次序 ↓

∴ $(138)_{10} = (10001010)_2$

链栈应用：数制转换

1. 算法思想如下：

（1）若N<>0，则将N％j取得的余数压入栈s中，执行（2）；

若N=0，将栈s的内容依次出栈，算法结束。

（2） 用N/j代替N

（3） 当N>0，则重复步骤（1）、（2）。

链栈应用：数制转换

2. 算法的实现：

```
void Conversion(int n)          // 将十进制数n转换为二进制数
{  linkstack s;
   int x;
   s.top=NULL;
  do
  { x=n%2;                                  // 除2取余
    n=n/2;
    stacknode  *p=new stacknode;  // 申请一个新结点
    p->next=s.top;                          // 修改栈顶指针
    s.top=p;
    s.top->data=x; }                        // 入栈
  while(n);
  printf("转换后的二进制数值为：");
  while(s.top)                              // 余数出栈处理
  {
      printf("%d",s.top->data);  // 输出栈顶的余数
      stacknode* p=s.top;          // 修改栈顶指针
      s.top=s.top->next;
      delete p;                   // 回收一个结点，C语言中用free p
          }
   }
```

链栈应用：子程序调用

在计算机程序设计中，子程序的调用及返回地址就是利用堆栈来完成的。

在C(或C++)语言的主函数对无参子函数的嵌套调用过程中，在调用子程序前，先将返回地址保存到堆栈中，然后才转去执行子程序。当子函数执行到return语句（或函数结束）时，便从栈中弹出返回地址，从该地址处继续执行程序。

链栈应用：子程序调用

例如：主函数调用子函数a()时，则在调用之前先将a函数返回地址压入栈中；在子函数a()中调用子函数b()时，又将b函数返回地址压人栈中；同样，在子函数b()中调用子函数c()时，又将c函数返回地址压人栈中。其调用返回地址进栈示意图如图3-5所示。

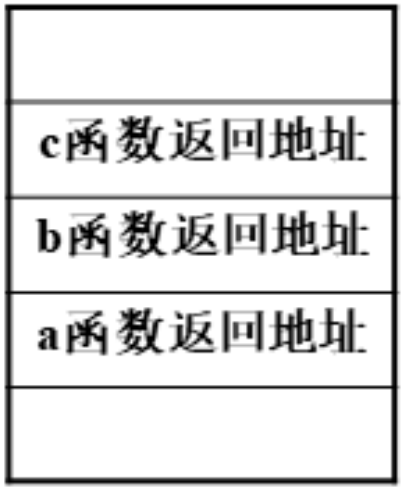


图3-5 无参函数嵌套调用返回地址进栈示意图

链栈应用：子程序调用

当执行完子函数c()以后，就从栈顶弹出c()函数返回地址，回到子函数b()；子函数b()执行完毕返回时，又从栈顶弹出b函数返回地址，回到子函数a()；子函数a()返回时，再在栈顶弹出a函数返回地址，回到主函数，继续执行主函数程序。无参函数嵌调用返回示意图如图3-6所示。

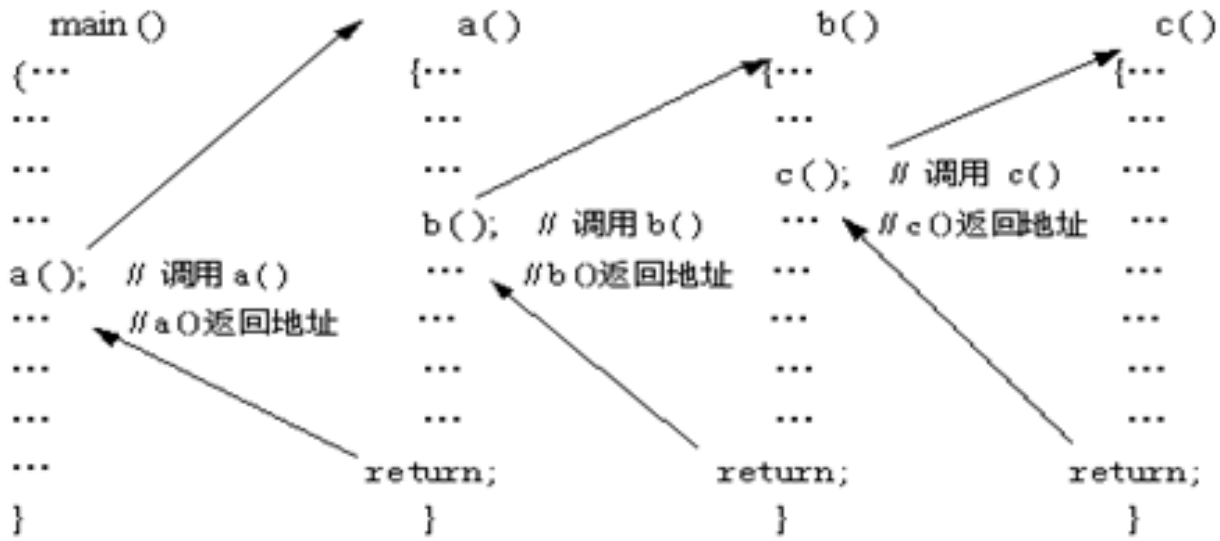


图3-6 无参函数嵌套调用返回示意图

链栈应用：递归调用

1．递归

一个直接调用自己，或者通过一系列调用语句间接地调用自己的函数称为递归函数。在程序设计中，有许多实际问题是递归定义的，使用递归的方法编写程序将使许多复杂的问题大大简化。所以，递归是程序设计中的一个强有力的工具

2．典型例子

（1）2阶斐波那契（Fibonacci）数列

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

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY

（2）阶乘函数

n!的定义为：

$$Fac(n)=\begin{cases}1 & n=0 \quad \text{// 递归终止条件}\\ n*Fac(n-1) & n>0 \quad \text{// 递归步骤}\end{cases}$$

根据定义不难写出相应的递归函数：

```
int  fac ( int n )
{  if  (n= =0)  return  1 ;
   else  return  (n* fac (n–1) );
}
```

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY

链栈应用：中断处理和现场保护

1.中断处理（Interrupt Processing）

在C++语言中，系统调用是通过中断来进行，中断调用示意图如图3-8所示。

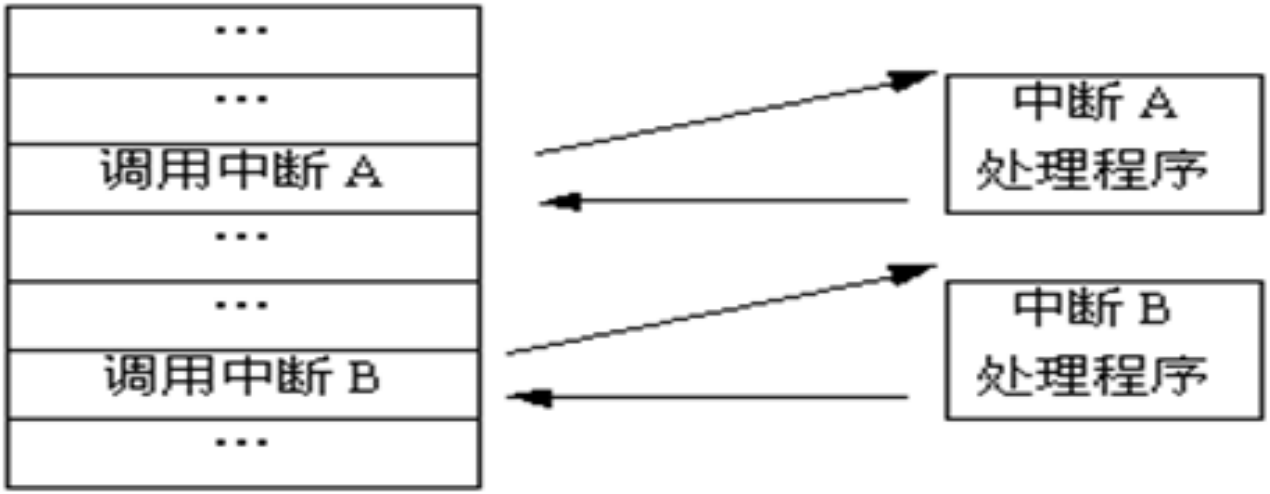


图3-8 中断调用示意图

如果把中断处理想象成函数调用，则中断处理程序可以看成被调用的函数。

2. 现场保护和恢复

执行中断时，微处理机有时必须对状态寄存器，累加器，以及相关的寄存器对进行现场保护（压栈）；中断处理完毕，则必须按后进先出的原则恢复现场（出栈）。

我们以汇编语言来说明现场保护和恢复的原理:

```
...                ；接受中断处理
PUSH AX            ；保护现场
PUSH BX
PUSH CX
PUSH BP
PUSHF              ；F状态寄存器进栈
...                ；中断处理
POPF               ；恢复现场，后进栈的先出栈
POP BP
POP CX
POP BX
POP AX
```

小结

（1）栈是一种运算受限制的线性表，它只允许在栈顶进行插入和删除等操作。

（2）栈的逻辑结构和线性表相同，数据元素之间存在一对一的关系，其主要特点是“后进先出”。

（3）栈的存储结构有顺序栈和链栈之分，要求掌握栈的C语言描述方法。

（4）重点掌握在顺序栈和链栈上实现：进栈、出栈、读栈顶元素、判栈空和判栈满等基本操作。

（5） 熟悉栈在计算机的软件设计中的各种应用，能灵活应用栈的基本原理解决一些综合性的应用问题。

GLOBALIZATION • INNOVATION • ENTREPRENEURSHIP • RESPONSIBILITY

单选题 1分

借助堆栈将中缀表达式A-(B-C/D)*E转换为后缀表达式，则该堆栈的大小至少为：

A 2

B 3

C 4

D 5

50

单选题 1分

设1、2、...、n-1、n共n个数按顺序入栈，若第一个出栈的元素是n，则第三个出栈的元素是：

A 3

B n-2

C n-3

D 任何元素均可能

51

单选题 1分

若用单向链表实现一个堆栈，当前链表状态为：1->2->3。当对该堆栈执行pop()、push(4)操作后，链表状态变成怎样？

（1）4->2->3　（2）1->2->4

A 只能是（1）

B 只能是（2）

C （1）和（2）都有可能

D （1）和（2）都不可能

52

单选题 1分

如果一堆栈的输入序列是aAbBc，输出为 abcBA，那么该堆栈所进行的操作序列是什么？ 设P代表入栈，O代表出栈。

A PPPOOPOPOO

B POOPPPOPOO

C POPPOPPOOO

D PPOPPOOOPO

53

实验3 栈子系统

1. 实验目的

（1） 掌握栈的特点及其描述方法。

（2）用链式存储结构实现一个栈。

（3）掌握建栈的各种等基本操作。

（4）掌握栈的几个典型应用的算法。

2. 实验内容

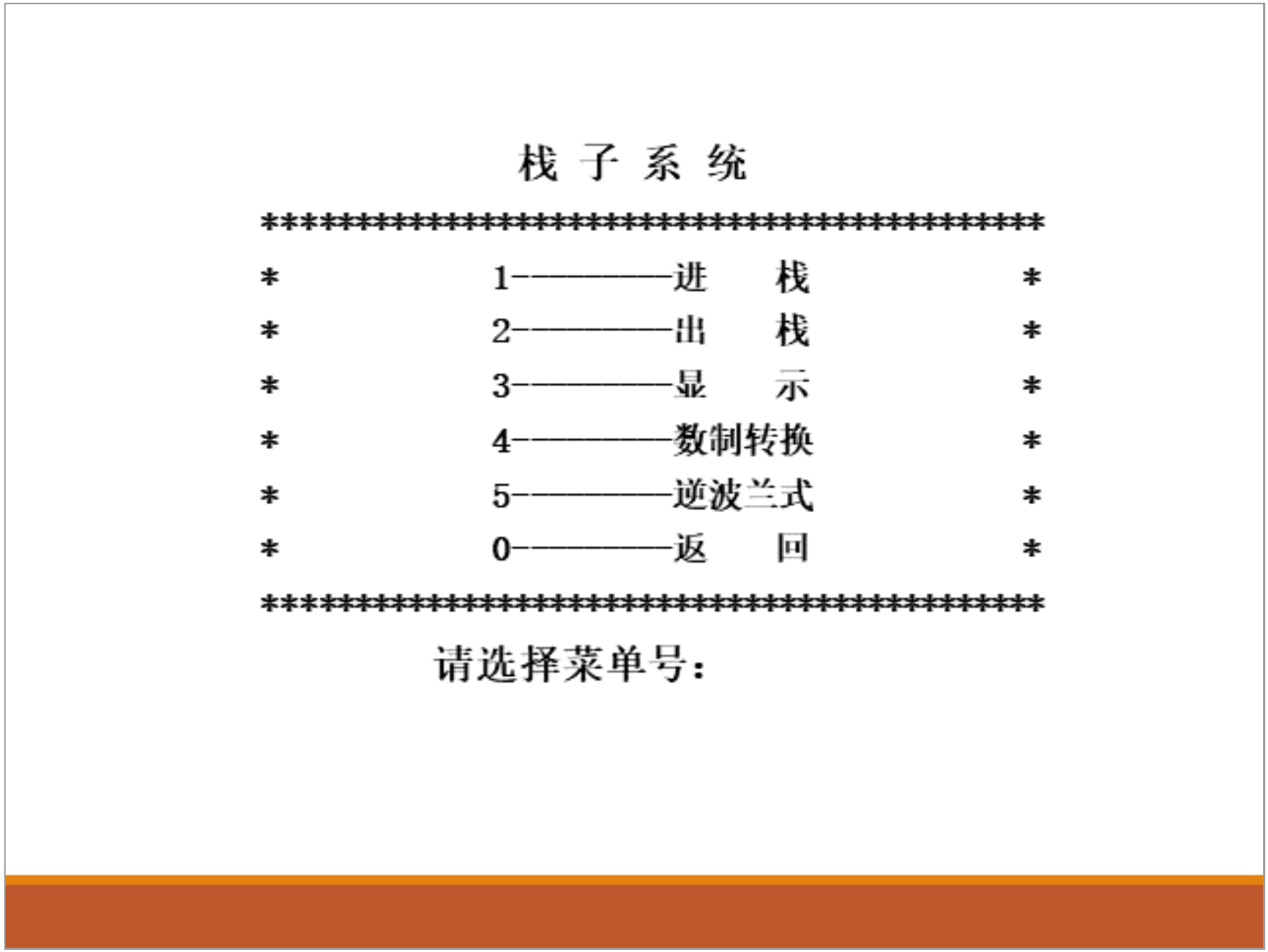
（1）设计一个字符型的链栈；

（2）编写进栈、出栈、显示栈中全部元素的程序；

（3）编写一个把十进制整数转换成二进制数的应用程序；

（4）编写一个把中缀表达式转换成后缀表达式（逆波兰式）的应用程序；

（5）设计一个选择式菜单，以菜单方式选择上述操作。


栈 子 系 统
****
* 1————进 栈 *
* 2————出 栈 *
* 3————显 示 *
* 4————数制转换 *
* 5————逆波兰式 *
* 0————返 回 *
****
请选择菜单号:

习题3

P22 3.2;

P23 3.7.