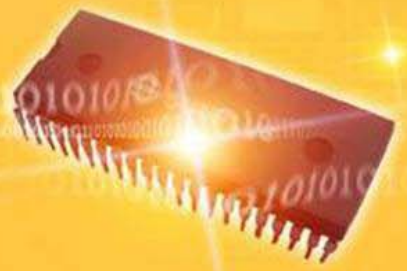
嵌入式系统工程师



栈和队列

引言

- 在现实编程中经常会遇到接收数据速度快而处理速度慢的情况，如果此时没有一个暂缓区来存储数据，那么很可能出现接收到的数据丢失的问题.
- 本章会用到2种方式来对以上问题的处理:

  栈、队列

大纲

- 栈的相关操作
- 队列的相关操作

大纲

- 栈的相关操作
- 队列的相关操作

栈的相关操作

- 栈(LIFO/FILO)

  限定只能在表的一端进行插入和删除的特殊的线性表.

- 这种线性表有两端:

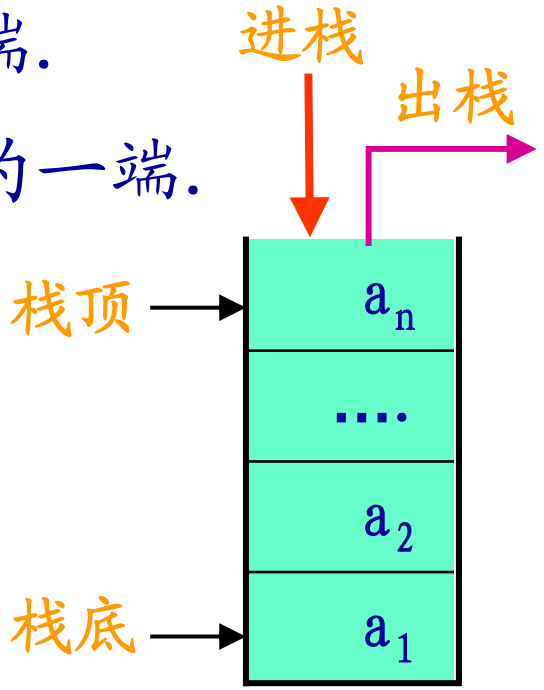
  栈顶(top): 允许插入和删除的一端.

  栈底(bottom):不允许插入和删除的一端.

  设栈s = ($a_1, a_2, \ldots a_n$);

  其中a_1是栈底元素，a_n是栈顶元素

  其原理示意图为:



栈的相关操作

- **栈的主要运算**
  - ① 进栈（压栈）
  - ② 出栈（弹栈）
  - ③ 判断栈满
  - ④ 判断栈空
- 凡是对数据的处理具有“后进先出”的特点，都可以用栈这种数据结构来操作.

栈的相关操作

➢ 存储结构-顺序栈

顺序栈利用一组连续的存储单元存放自栈底到栈顶的数据元素.通常可用一维数组设计栈.

顺序栈

➢ 顺序栈

设置一个简单变量top指示栈顶位置，top称为栈定指针.

约定：top始终指向新数据元素将存放的位置.

➢ 栈的空满情况:

top=0　　表示栈空;

top=stacksize　　表示栈满;

stacksize　　是栈的最大容量

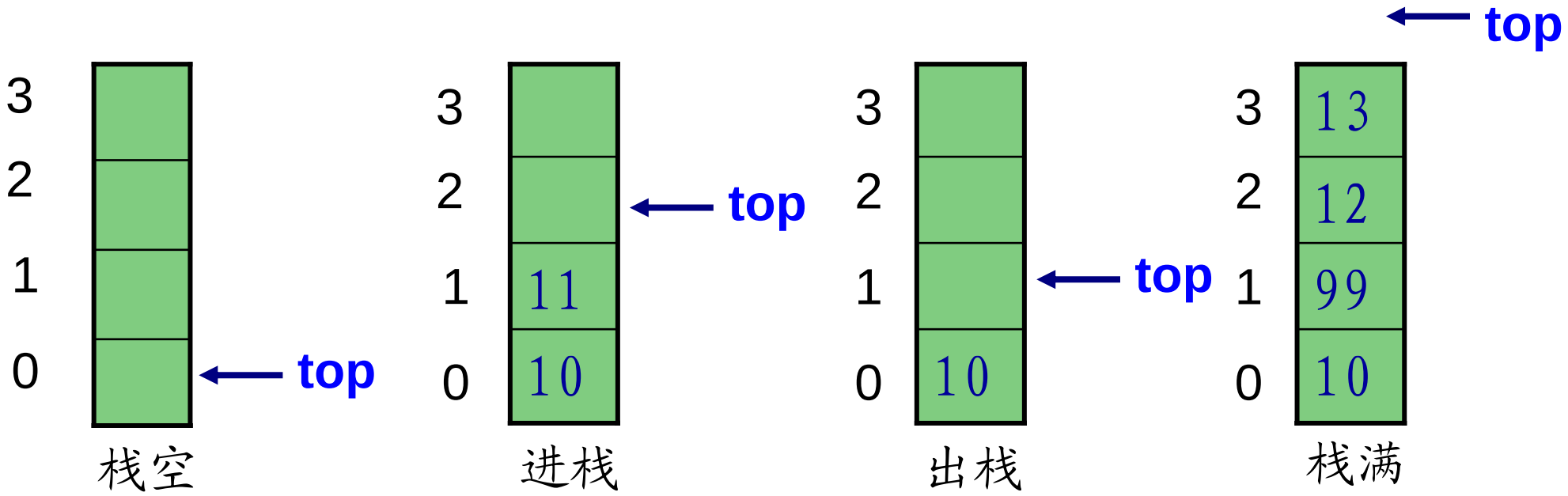
➢ 栈的溢出情况:

① 当栈满时(top=stacksize),若还有要进栈，称为“上溢”.

② 当栈满时(top=0)，若还要退栈，称为“下溢”.

无论上溢或下溢，程序中都要显示信息，以便处理.

顺序栈



s[top] = num;
top++;

top--;
num = s[top];

注意：

① top只是一个变量来表示当前要存放数据的位置.

② 进栈：先存放数据，再对top++.

③ 出栈：先对top--，在取数据.

④ 当top指向位置超出可以存放空间时，代表栈满.

顺序栈应用

```
int main()
{
    static int s[3];
    int top=0,result,y;
    int i;

    result=push(s,11,&top);      //将11压进栈中
    result=push(s,22,&top);      //将11压进栈中
    result=push(s,33,&top);      //将11压进栈中
    printf("top=%d\n",top);

    for (i=0;i<3;i++)
    {
        result=pop(s,&y,&top);  //将11从栈中弹出
        printf("top=%d,y=%d\n",top,y);
    }

}
```

顺序栈应用

```
int push(int s[],int x,int *ptop)
{
    int top;
     top=*ptop;                    //ptop是指针变量；*ptop获得实际栈顶指针
    if ( top== 3 )                 //栈满
     {
        printf("overflow\n");
        return 0;
    }
    else
    {
        s[top]=x;
        top++;
        *ptop=top;                 //实际栈顶指针加1，返回到调用函数处
        return 1;
    }
}
```

顺序栈应用

```
int pop(int s[],int *py,int *ptop)
{
    int top;
    top=*ptop;                          //ptop是指针变量；*ptop获得实际栈顶指针
    if (top==0)                         //栈空
    {
       printf("stack empty\n");
       return 0;
    }
   else
   {
       --top;
       *py=s[top];                      //实际栈顶指针减1，返回到调用函数处
       *ptop=top;
        return 1;
   }
}
```

大纲

- 栈的相关操作
- 队列的相关操作

队列的相关操作

- **队列 (FIFO)**

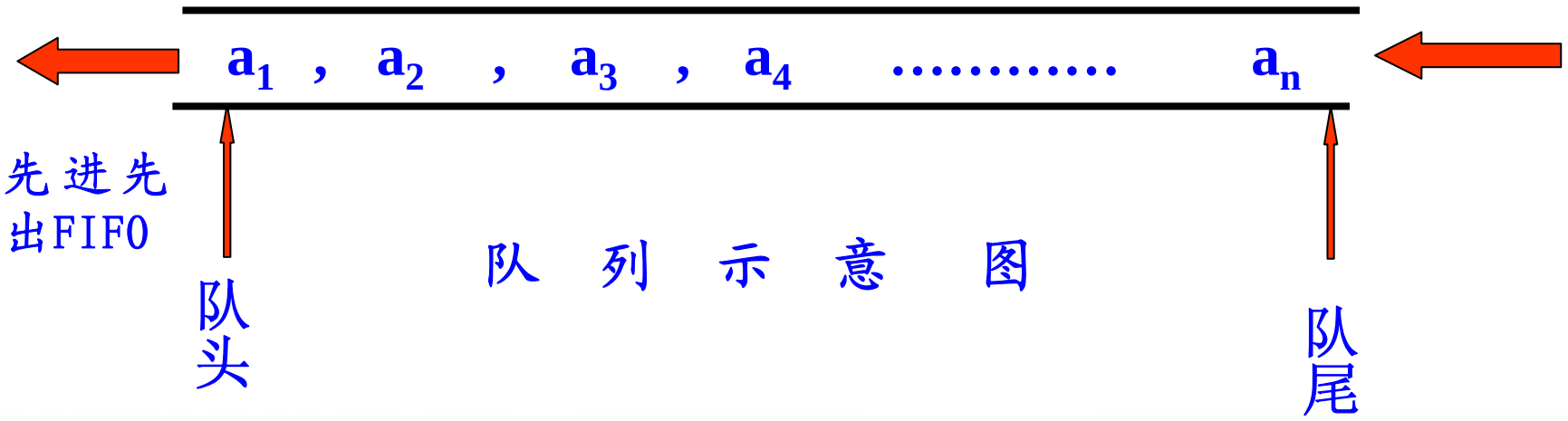
  限定在一端插入，另一端删除的一种特殊线性结构.

- **队列的两端**

  ① 允许插入数据的一端叫做队尾.

  ② 允许数据离开的一端叫做对头.

  如下图



队 列 示 意 图

队列的相关操作

- 队列的主要运算

  ① 设置一个空队列.

  ② 插入一个新的队尾元素，称为入队.

  ③ 删除队头元素，称为出队.

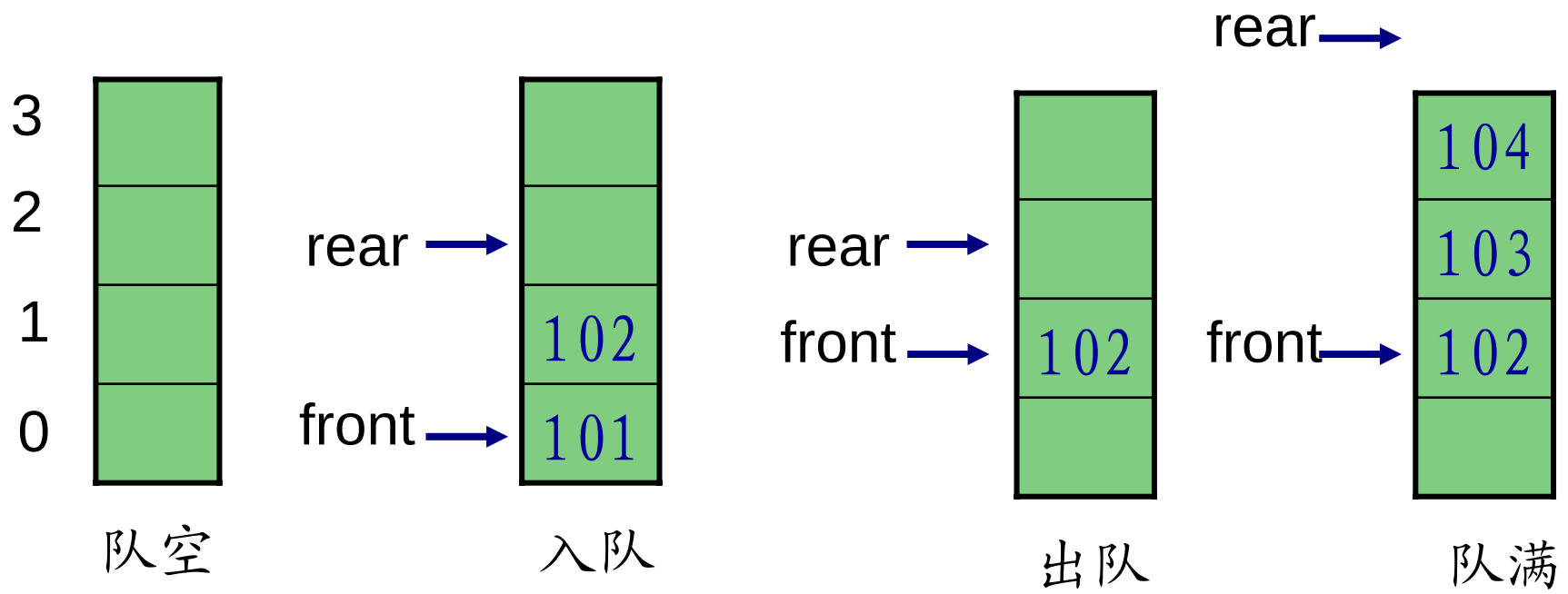
  ④ 读取队头元素.

- 凡是对数据的处理具有“先进先出”的特点，都可以用队列这种数据结构来操作.

队列的相关操作

➢ 队列的存储结构-顺序存储

① 线性队列



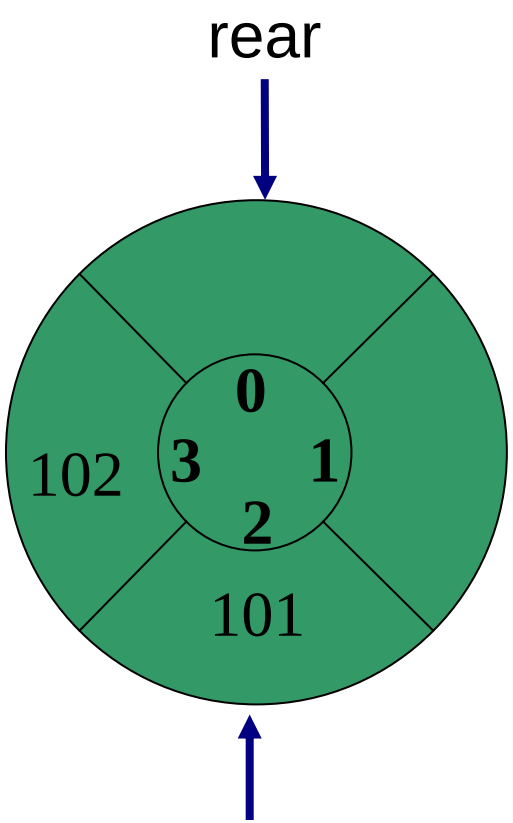
rear=front=0（队空）

队列的相关操作

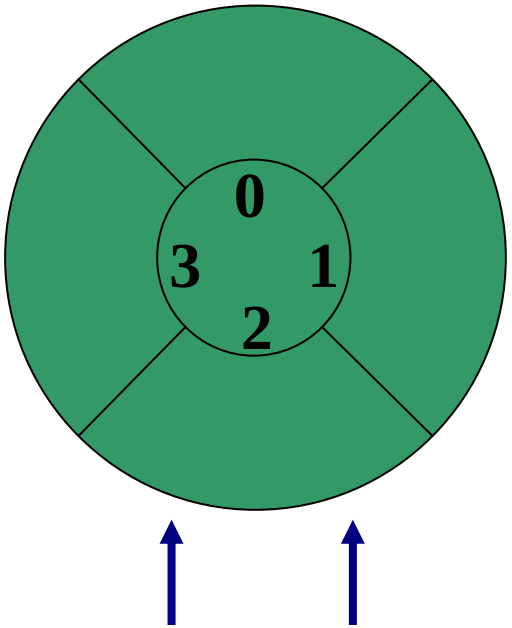
② 循环队列

队满条件:

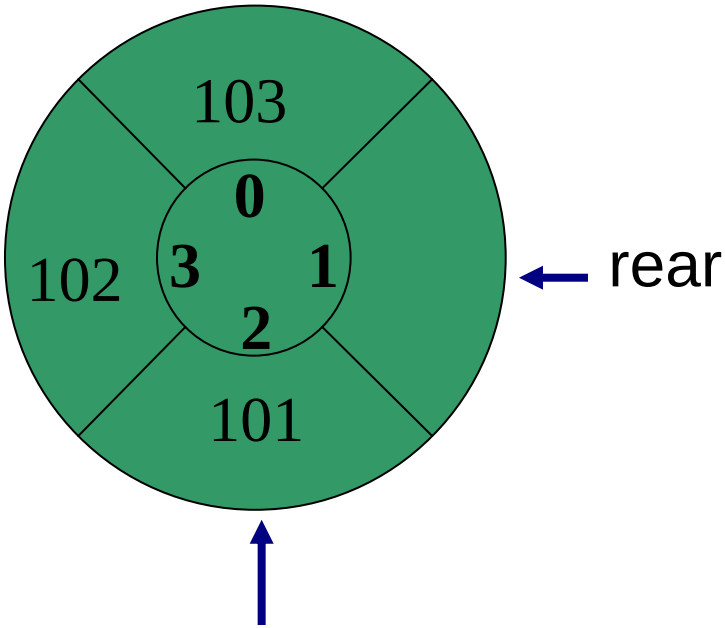
(rear+1)%MAX=front



一般情况



队空



队满

循环队列应用

```
int main()
{
    static int s[MAX];

    int rear = 0,front = 0;
    int b=0;
    EnQueue(s,11,&rear,&front);      //入队操作

    DeQueue(s,&b,&rear,&front);      //出队操作

    printf("%d",b);
}
```

循环队列应用

```
int EnQueue(int *Q,int x,int *pf,int *pr)
{
    int front,rear;
    front = *pf;
    rear=*pr;
    if ( (rear+1) % MAX == front )  //判断队满
        return 0;
    else
    {
        Q[rear]=x;                  //将数据存入队列
        rear = ( rear+1 ) % MAX;    //队尾指针移位
         *pr=rear;                  //保存队尾指针
         return 1;
    }
}
```

循环队列应用

```
int DeQueue(int *Q,int *py,int *pf,int *pr)
{
    int front,rear;
    front=*pf;
    rear=*pr;
    if (front==rear)                    //判断队空
        return 0;
    else
    {
        *py=Q[front];                   //将数据读出
        front=(front+1) % MAX;          //队头指针移动
        *pf=front;                      //保存队头指针
        return 1;
    }
}
```

凌阳教育
www.sunplusedu.com

值得信赖的教育品牌

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com