下面是完全自己写栈的基本功能的主要工作

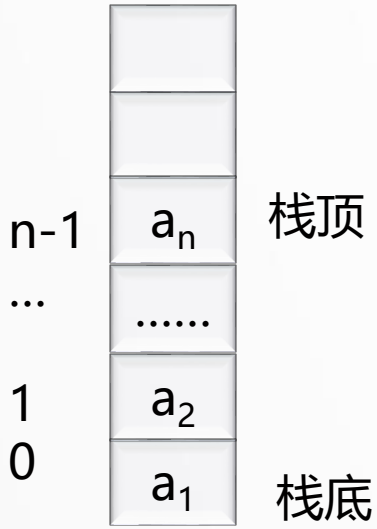
2.2.2 栈的顺序存储

类型定义

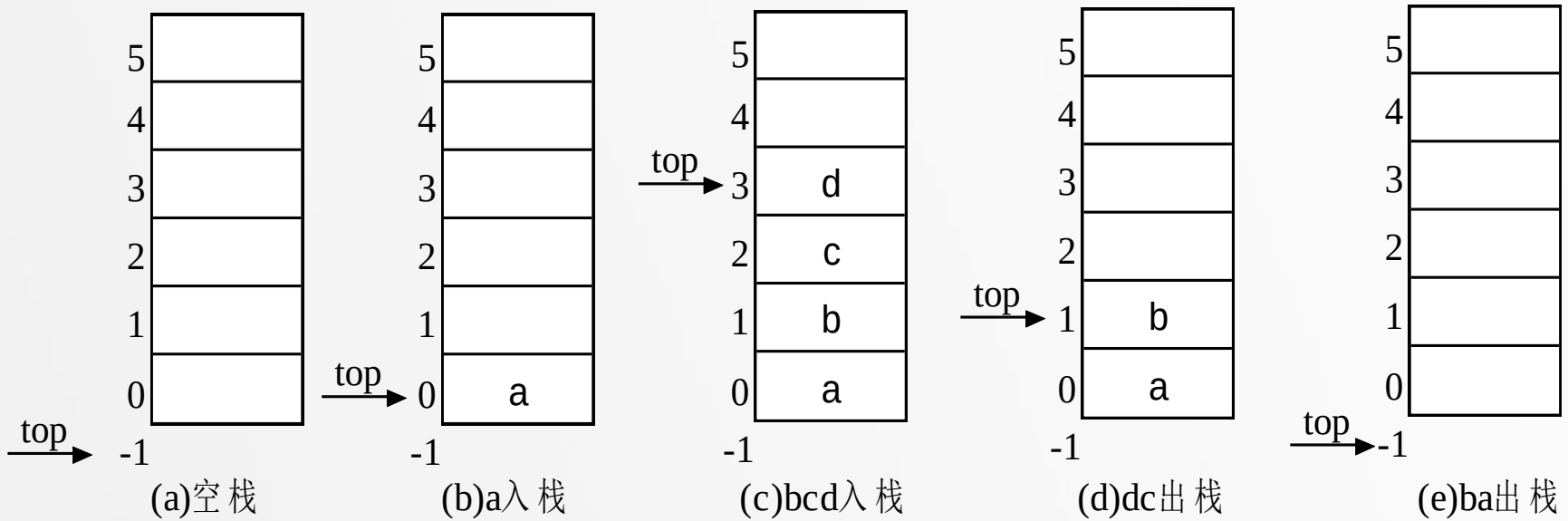
```
typedef struct stack {
    int top;//就是线性表顺序存储结构的length
    StackEntry * elem;//动态分配空间大小为stack_size
    int stack_size;
} Stack,*StackPtr;
```

域elem[0.. stack_size-1]用于存放数据元素

约定top用于存放栈顶元素的位置，
top = -1表示空栈，
top=stack_size-1表示栈满

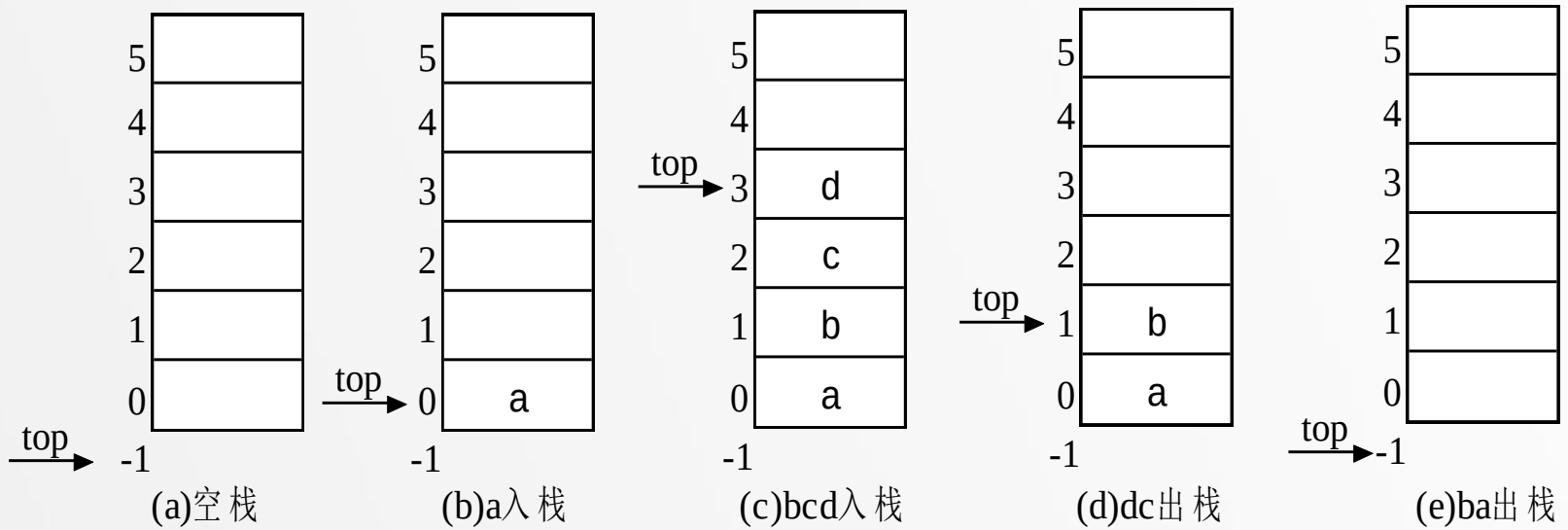


顺序栈



- 入栈时top指针加1；出栈时top指针减1

顺序栈



- 溢出
  - 顺序栈的数据元素空间大小是预先分配
  - 当空间全部占满后再入栈产生的溢出称为“上溢”;
  - 当栈为空时再出栈也将产生的溢出称为“下溢”

顺序栈**入栈**操作的实现

```
Status   Stack_Push(StackPtr s, StackEntry item){
    Status  outcome = success;
if ( s-> top==MAXSTACK-1)
        outcome = overflow;  /* 栈满则上溢 */
    else{
        s->top++;
        s->entry[s->top] = item;  /*数据元素放入top位置 */
    }
    return outcome;
}
```

顺序栈**出栈**操作的实现

```
Status  Stack_Pop(StackPtr s, StackEntry *item){
    Status  outcome = success;
    if ( s-> top== -1)
        outcome = underflow;      /* 栈空则下溢 */
          else
        *item = s->entry[s->top--]; /*将top所指数据元素放入item，top再减1 */
    return outcome;
}
```

顺序栈**取栈顶元素**操作的实现

```
Status  Stack_Top(StackPtr s, StackEntry *item){
   Status outcome = success;
   if (Stack_Empty(s))
        outcome = underflow; /* 栈空则下溢出 */
   else
       *item = s->entry[s->top]; /*取出数据，top指针不变 */
   return outcome;
}
```

多个顺序栈空间共享