

链表与顺序表(基础线性表)

顺序表 → 有额外元素的数组, 连续存储

如:

0	1	2	3	4	5	6	7	8
①	②	③	④	⑤				

1、size = 9 → 大小
2、count = 5 → 存Data个数

特点: 读取Data速度快 $O(1)$

结构操作: 插入与删除

[] :	右移多位元素	左移多位补空缺
size :	+	-1
count :	+1	-1

```
typedef struct vector
{
    int size, count;
    int *data;
}vector;
```

1> typedef 为叫 vector 的结构体
取一个叫 vector 的新类型名

2> 不知大小的数组 → *data
来方便扩容

① 初始化代码 — C:

```
// 初始化一个空的顺序表
vector *init(int size)
{
    vector* v = (vector*)malloc(Size: sizeof(vector));
    v->size = size;
    v->count = 0;
    v->data = (int*)malloc(sizeof(int)*size);
    return v;
}
```

1> 大小由初始化输入决定 → 参数
2> 先开辟v的空间再开辟顺序表中数组的空间

存储上限为 size

② 销毁代码 — C:

```
// 销毁一个顺序表
void destroy(vector *v)
{
    free(v->data);
    free(v);
}
```

1> 逐层销毁的方法
2> 从内到外开始销毁

malloc 与 free 要一一对应!!!

③ 插入操作 → C:

// 插入一个数据

```
int insert(vector *v,int pos,int val)
{
    if(v->count == v->size)
    {
        printf(format:"Your vector is full\n");
        return 0;
    }
    for(int i = v->size; i > pos; i--)
    {
        v->data[i] = v->data[i-1];
    }
    v->data[pos] = val;
    v->count++;
    printf(format:"You have inserted %d to %d\n",v->data[pos],pos);
    return 1;
}
```

1> 插入合法性验证必不可少

2> 逆序索引移位

3> count 艾属性值必须变

④ 删除操作 → C:

// 删除操作

```
int erase(vector *v,int pos)
{
    if(pos<0||pos>v->count)
    {
        printf(format:"Your position is not right\n");
        return 0;
    }
    for(int i = pos; i < v->size; i++)
    {
        v->data[i] = v->data[i+1];
    }
    v->count--;
    printf(format:"You have erased %d in %d\n",v->data[v->size-1],pos);
    return 1;
}
```

1> 顺序遍历前移

逆序遍历后移

⑤ 插入+扩容 → C:

// 双倍扩容操作

int expend(vector *v)

```
{
    if(v == NULL)return 0;
    if(v->count == v->size)
    {
        v->data = (int *)realloc(v->data, NewSize:sizeof(int)*v->size*2);
    }
    v->size = v->size * 2;
    return 1;
}
```

对 realloc 第三个机制不友好有 bug
↓ 修改
int *p = (int *)realloc(v->data, sizeof(int)*v->size*2);
if(p == NULL) return 0;
else v->data = p;

1> 利用 realloc 扩容 data 大小

同时同步修改 size 大小

2> 在满的时候 insert_plus
启动 expend 函数

realloc 机制:

// 配套插入操作

int insert_plus(vector *v,int pos,int val)

```
{
    if(v->count == v->size)
    {
        expend(v);
    }
    for(int i = v->size; i > pos; i--)
    {
        v->data[i] = v->data[i-1];
    }
    v->data[pos] = val;
    v->count++;
    printf(format:"You have inserted %d to %d\n",v->data[pos],pos);
    return 1;
}
```