2.1 向量-接口与实现

cpp实现链接	https://dsa.cs.tsinghua.edu.cn/~deng/ds/src_link/vector/vector.h.htm
备注	关于Vector模板的构造以及复制函数的实现，需要注意！
学习日	
视频起始	https://www.bilibili.com/video/av75509584?p=34
讲义	02.Vector.A.Interface_Implementation.pdf

▼ 02-A-1 接口与实现

- 目标
  - 实现一个数据结构
  - 实现（搜索与排序）算法，对外接口可以更加高效地实现
- 抽象数据类型 vs 数据结构
  - 抽象数据类型 = 数据模型 + 一系列操作
  - 数据结构 = 特定语言 + 算法
- 将Vector/List直接抽象为一个数据类型

  ADT（即为接口），两类关联者：实现/应用者

▼ 02-A-2 向量ADT

- C/C++语言中，数组A[]中的元素与[0, n)内的编号一一对应，连续存储
  - 反之，每个元素由一个非负编号唯一指代，可以直接访问
  - A[i]物理地址 = A + i * s，s为单个元素占用空间 → 由于是物理地址由线性方程确定，因此被称为线性数组（linear array)
- 向量ADT看作数组的一种抽象与繁华，由一组元素按线性次序封装而成

  各元素与[0, n)内的**秩（rank）**意义对应 → call-by-rank

  元素不限于基本类型；操作，管理更简化、统一与安全；从而实现复杂数据结构的定制与实现

- 向量ADT的操作接口

向量ADT接口

操作	功能	适用对象
size()	报告向量当前的规模（元素总数）	向量
get(r)	获取秩为r的元素	向量
put(r, e)	用e替换秩为r元素的数值	向量
insert(r, e)	e作为秩为r元素插入，原后继依次后移	向量
remove(r)	删除秩为r的元素，返回该元素原值	向量
disordered()	判断所有元素是否已按非降序排列	向量
sort()	调整各元素的位置，使之按非降序排列	向量
find(e)	查找目标元素e	向量
search(e)	查找e，返回不大于e且秩最大的元素	有序向量
deduplicate(), uniquify()	剔除重复元素	向量/有序向量
traverse()	遍历向量并统一处理所有元素	向量

▼ 02-A-3 操作实例

disordered() 查看数组中相邻元素（增大为正序）降序的个数，如43.74.96

find() 返回rank，若不存在，则返回-1

sort() 增量排序

search(e) 返回不超过e对应的rank；若查找小于全局最小，则返回-1；若重复出现，取rank最大的数

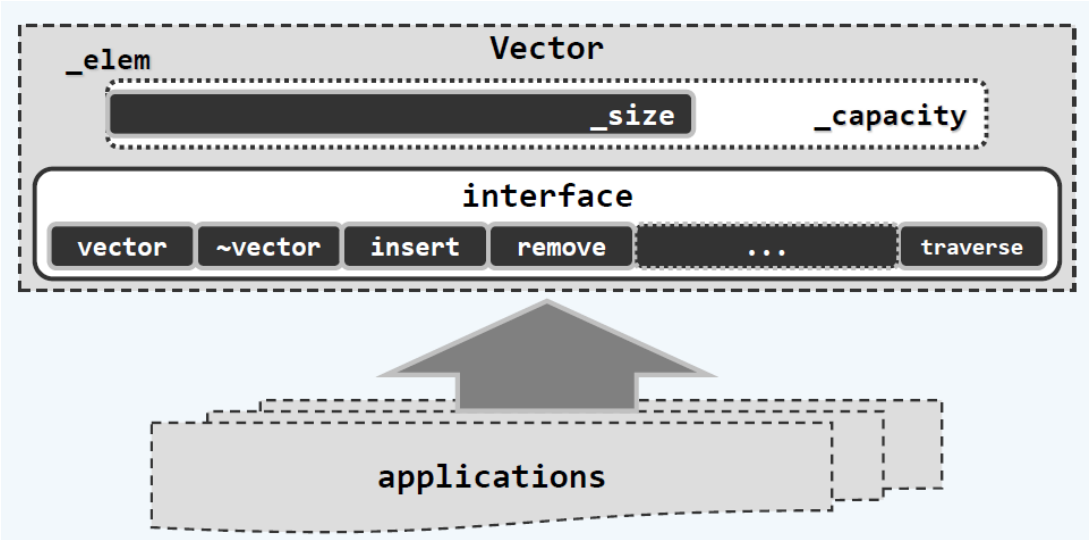
操作	输出	向量组成（自左向右）
初始化		
insert(0, 9)		9
insert(0, 4)		4 9
insert(1, 5)		4 5 9
put(1, 2)		4 2 9
get(2)	9	4 2 9
insert(3, 6)		4 2 9 6
insert(1, 7)		4 7 2 9 6
remove(2)	2	4 7 9 6
insert(1, 3)		4 3 7 9 6
insert(3, 4)		4 3 7 4 9 6
size()	6	4 3 7 4 9 6

操作	输出	向量组成（自左向右）
disordered()	3	4 3 7 4 9 6
find(9)	4	4 3 7 4 9 6
find(5)	-1	4 3 7 4 9 6
sort()		3 4 4 6 7 9
disordered()	0	3 4 4 6 7 9
search(1)	-1	3 4 4 6 7 9
search(4)	2	3 4 4 6 7 9
search(8)	4	3 4 4 6 7 9
search(9)	5	3 4 4 6 7 9
search(10)	5	3 4 4 6 7 9
uniquify()		3 4 6 7 9
search(9)	4	3 4 6 7 9

▼ 02-A-4 构造与析构

- 结合已有操作，在C/C++中构建Vector模板类

```
typedef int Rank; //秩
#define DEFAULT_CAPACITY  3 //默认的初始容量（实际应用中可设置为更大）
template <typename T> class Vector { //Vector模板类
  protected: Rank _size; int _capacity;  T* _elem; //内部函数：规模、容量、数据区
  public:
    // 构造函数
    // 析构函数
    // 只读访问接口
    // 可写访问接口
    // 遍历接口
};
```



直接可以按照接口规范进行实现！

- 构造方法

  默认方法是直接new一个地址，但是因为没有额外内容，所以规模_size = 0

  还有其他使用copyFrom的方法实现复制

  ~Vector()则是删除，直接释放空间

```
❖Vector(int c = DEFAULT_CAPACITY)
    { _elem = new T[_capacity = c]; _size = 0; } //默认
❖Vector(T const * A, Rank lo, Rank hi) //数组区间复制
    { copyFrom(A, lo, hi); }
  Vector(Vector<T> const& V, Rank lo, Rank hi)
    { copyFrom(V._elem, lo, hi); } //向量区间复制
  Vector(Vector<T> const& V)
    { copyFrom(V._elem, 0, V._size); } //向量整体复制
❖~Vector() { delete [] _elem; } //释放内部空间
```

▼ 02-A-5 复制

copyFrom()函数实现

```
❖template <typename T> //T为基本类型，或已重载赋值操作符'='
  void Vector<T>::copyFrom(T const * A, Rank lo, Rank hi) {
     _elem = new T[_capacity = 2*(hi - lo)]; //分配空间
     _size = 0; //规模清零
     while (lo < hi) //A[lo, hi)内的元素逐一
        _elem[_size++] = A[lo++]; //复制至_elem[0, hi - lo)
  }  //O(hi - lo) = O(n)
```

分配两倍的空间 2 * (hi - lo)，**预留空间，避免扩容而打断计算过程**！然后清零以及，循环逐一赋值