02-A3

向量

抽象数据类型：模板类

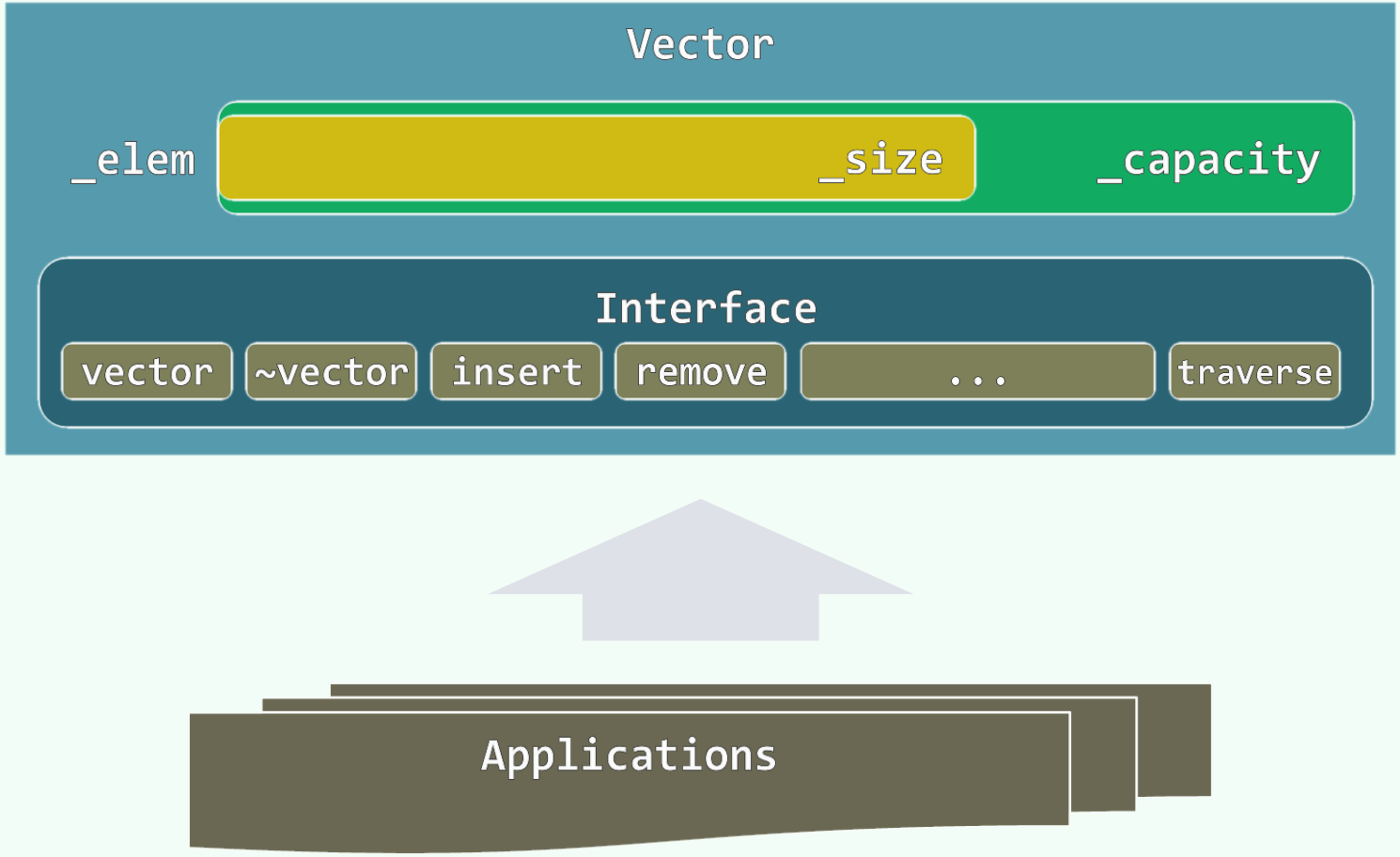
官职须由生处有，文章不管用时无

堪笑翰林陶学士，年年依样画葫芦

邓俊辉

deng@tsinghua.edu.cn

```
template <typename T> class Vector { //向量模板类
   private: Rank _size; int _capacity; T* _elem; //规模、容量、数据区
   protected:
      /* ... 内部函数 */
   public:
      /* ... 构造函数 */
      /* ... 析构函数 */
      /* ... 只读接口 */
      /* ... 可写接口 */
      /* ... 遍历接口 */
};
```



构造 + 析构

- `#define DEFAULT_CAPACITY  3 //默认初始容量（实际应用中可设置为更大）`
- `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 ) //向量区间复制`
  `{ copyFrom( V._elem, lo, hi ); }`
  `Vector( Vector<T> const& V ) //向量整体复制`
  `{ copyFrom( V._elem, 0, V._size ); }`
- `~Vector() { delete [] _elem; } //释放内部空间`

基于复制的构造

```
❖ 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，逐一复制至_elem[0, hi - lo)
        _elem[ _size++ ] = A[ lo++ ];
  }  //O(hi - lo) = O(n)
```