OOP 第九周作业文档

2019010175 孔瑞阳 土木 92

一、项目信息

1、功能说明

(1)

输入第一行一个正整数 n，表示排序的数的个数。
输入第二行 n 个整数，表示需要排序的数。

输出第一行不去重排序的结果，用 sort()实现。
输出第二行　去重排序的结果，用 set　实现。

(2)

模仿 vector 实现了一个 MyVector，具有以下主要功能：

MyVector() {}	默认构造
MyVector(size_t n, const T& value = T())	赋值构造
MyVector(MyVector<int>& vec)	拷贝构造
MyVector(iterator begin, iterator end)	复制构造
~MyVector()	析构
MyVector<T>& operator=(const MyVector<T>& vec)	赋值
void swap(MyVector<T>& vec)	交换
iterator begin()	最前面的元素的迭代器
iterator end()	最后面的元素的后面一个迭代器
size_t capacity()	容量
size_t size()	长度
void reserve(size_t n)	修改容量
void resize(size_t n, const T& value = T())	修改长度
void clear()	清空
bool empty()	判断是否为空
T& at(size_t pos)	访问某个位置的引用（成员函数形式）
T& operator [](size_t pos)	访问某个位置的引用（数组形式）
T& front()	第一个位置的引用
T& back()	最后一个位置的引用
void insert(iterator pos, const T& x)	在某一位置插入
void push_back(const T& x)	在最后一个位置插入
iterator erase(iterator pos)	删除某个位置
void pop_back()	删除最后一个位置

2、测试环境

CPU	Intel(R) Core(TM)i7-9750H CPU @ 2.6Ghz 6 核 12 线程
GPU	NVIDIA GeForce RTX2070
RAM	DDR4 16G+16G
Operating System	Microsoft Windows 版本 1909
Compiler	MSVC++ 14.24

二、验证

1、排序的验证

(1) 手动验证

分为没有重复数据和有重复数进行验证。

数据	不去重排序（sort）	去重排序（set）
9 6 6 4 7 5 8 7 1 2	1 2 4 5 6 6 7 7 8 9	1 2 4 5 6 7 8 9
10 10 10 10 10 10 10 10	10 10 10 10 10 10 10 10	10
0 9 7 3 1 4 6 8 2 5	0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9

(2) 对拍验证

用 unique 对用 sort 进行不去重排序之后的数组进行去重，并与用 set 进行去重排序之后的结果进行比较，如果元素个数相同、结果相同的话，那么就可以认为两种排序算法都是正确的。

进行了 10 分钟的对拍，大概进行了 600 次验证，都没有出现错误，可以基本认为两种程序都是正确的。

2、MyVector 的验证

操作	MyVector 中的元素
push_back(0)...push_back(9)	0 1 2 3 4 5 6 7 8 9
5 次 pop_back	0 1 2 3 4
拷贝构造 V(v)	0 1 2 3 4
erase(v.begin())	1 2 3 4
insert(v.end() - 2, 5)	1 2 5 3 4
V.front() = 5, V.back() = 3, V[1] = 4, V.at(3) = 4	5 4 5 4 3
empty(), clear(), empty()	(FALSE TRUE)(0 1)
复制构造 w(v.begin() + 1, v.end() - 1)	1 2 3
resize(5,4)	1 2 3 4 4
resize(4)	1 2 3 4
V.swap(w)	1 2 3 4
v = V	1 2 3 4
默认构造 W	
赋值构造 WW(6, 6)	6 6 6 6 6 6

另外，析构、size()、capacity()、reerve()，也都在上述操作和输出的过程中涉及了。
至此，所有操作的初步正确性验证完毕。

3、MyVector 和 vector 的效率比较

选取了 push_back, pop_back, insert, erase, 赋值构造这几个操作进行了比较。
分 不开编译优化，和 O2 编译优化进行比较。

操作	MyVector	MyVector (O2)	vector	Vector (O2)
10^7 次 push_back	1.403s	1.229s	3.529s	3.326s
10^7 次 pop_back	0.404s	0.349s	1.58s	1.34s
100 次插入 10^4 个数	5.799s	6.871s	2.336s	3.115s
100 次删除 10^4 个数	5.505s	6.838s	1.676s	2.277s
100 次构造 10^6 个数	0.763s	0.733s	13.058s	13.595s

可以发现，最基础的 push_back 和 pop_back 操作，MyVector 的时间运行效率是 vector 的大概 2.5 倍，并且，O2 编译优化有一定的优化效果。

但是对于 insert 和 erase 操作，MyVector 的效率大概只有 vector 的 80%，并且发现 O2 编译优化似乎出现了比较严重的负优化……

对于赋值构造 MyVector(size_t n, const T& value = T())操作，MyVector 的速度大概是 vector 的 20 倍，O2 优化的效果不明显，或者有较小的负优化。

总的来说，对于构造、push_back 和 pop_back，手写的 MyVector 的运行效率较好，并且 O2 编译优化有一定的效果。对于 insert 和 erase 操作，虽然 vector 比 MyVector 快速，但是在实际运用当中，因为 insert 和 erase 都是线性复杂度，如果操作数多的话就会选择平衡树类的数据结构。所以，总体来说，MyVector 的效率要比 vector 更高。