

**数据结构实验报告**



实验题目 1.对比数组和向量的使用方法

2.设计函数，实现删除向量中 所有重复元素的功能。

学 号 201806021511

姓 名 姜义杰

班 级 网工1801

提交日期 2020.10.12

# 题目一

1. 实验内容：

设计两个函数实现冒泡排序算法，待排序数据分别以数组和向量作为参数传递到排序算法子函数，要求通过该子函数对数据进行排序，排序结果返回到主函数，不得修改破坏原始待排数据；

二、程序源代码

#include"d\_vector.h"

#include<iostream>

#include<cstdio>

using namespace std;

int\* bubblesort(int\* a, int n)

{

int\* b = new int[n];

memcpy(b, a, n\*sizeof(int));

int end = n - 1;

while(end!=1)

{

bool flag = false;

for (int j = 1,i=0; j <= end;i++, j++)

{

if (b[i] > b[j])

{

int tmp = b[i];

b[i] = b[j];

b[j] = tmp;

flag = true;

}

}

end--;

if (flag == false)break;

}

return b;

}

template <typename T>

miniVector<T> bubblesort(miniVector<T>& a)

{

miniVector<T> b(a);

int end = b.size() - 1;

while (end != 1)

{

bool flag = false;

for (int j = 1, i = 0; j <= end; i++, j++)

{

if (b[i] > b[j])

{

T tmp = b[i];

b[i] = b[j];

b[j] = tmp;

flag = true;

}

}

end--;

if (flag == false)break;

}

return b;

}

template<class T>

void print(miniVector<T> a)

{

for (int i = 0; i < a.size(); i++)

{

cout << a[i] << " ";

}

cout << endl;

}

void print(int \* a,int n)

{

for (int i = 0; i < n; i++)

{

cout << a[i] << " ";

}

cout << endl;

}

int main()

{

miniVector<int> a(10);

int b[10];

//freopen("in.txt", "r", stdin);

for (int i = 0; i < 10; i++)

{

cin >> a[i];

b[i] = a[i];

}

int \*tmp1=bubblesort(b, 10);

print(tmp1, 10);

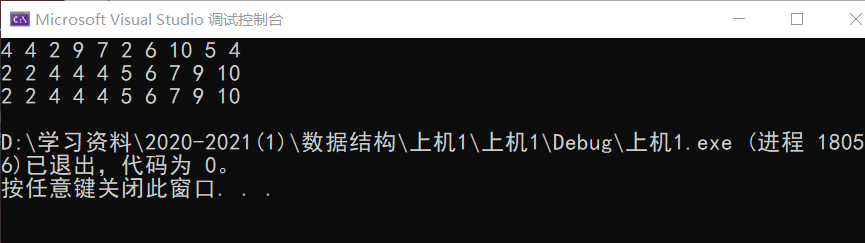
miniVector<int> tmp2=bubblesort(a);

print(tmp2);

return 0;

}

三、程序运行结果截图



四、讨论分析

1. **运行结果分析**

结果输出为输入的十个数字从小到大排序，与预期结果一致，答案正确。

1. **题目分析**

要求将数组和向量作为参数传递到子函数里面进行冒泡排序，并且不修改破坏原始数据，所以需要在函数内部新开辟一个临时数组和向量，排好序后返回到主函数中。

其中冒泡排序（升序）的思路为从左到右，两两比较，如果左边比右边大，则交换两个元素的位置。所以每轮比较都可以将最大的元素移动至最右边，每一轮遍历，右边有序区元素就增加一个，用一个变量记录有序区最左端的元素位置，当变量移动至最左端，排序结束。

在此函数中数组和向量的主要区别是数组的大小需要通过参数传递进函数里，而向量可以直接通过类函数size()获得。

1. **程序设计思路**

设计两个重载函数bubblesort，第一个对数组排序，参数表为一个数组指针int \*a和数组大小int n，返回值为int \*。第二个对向量模板排序，参数表为miniVector<T>& a，返回值为miniVector<T>。

对于int\* bubblesort(int\* a, int n)，首先动态分配一个大小为n的数组b然后将数组a的内容拷贝至b中,然后对b进行冒泡排序，用一个布尔类型变量flag记录一轮遍历中是否有元素交换，若没有交换，说明已经有序，直接退出循环。用变量end记录有序区位置，当end为1时，代表排序完毕，退出循环，因为右边n-1个元素已经有序，最左边的元素肯定也排好序了。之后将排好序的数组指针b返回至主函数中。

对于函数miniVector<T> bubblesort(miniVector<T>& a)，其内容与第一个函数基本类似，有一点不同是向量的元素个数可以直接通过类函数size()获得，而不用从参数表传值。

另外只需要在主函数中声明一个数组指针和一个向量，再用函数返回值赋值即可。

1. **数学原理说明**

由于冒泡排序实现需要两重循环，所以时间复杂度为。

特殊情况下，如待排序数组已经有序，则只需要 的时间复杂度。

# 题目二

1. 实验内容：

设计函数，实现删除向量中所有重复元素的功能。要求：尽可能避免数据大量移动，同时保持原始数据顺序，并分析算法的时间复杂度。

二、程序源代码

#include"d\_vector.h"

#include<iostream>

using namespace std;

template <typename T>

miniVector<T> unique\_NEW(miniVector<T> a)

{

miniVector<T> tmp;

for (int i = 0,j=0; i < a.size(); i++)

{

bool flag = true;

for (j = 0; j < tmp.size(); j++)

{

if (a[i] == tmp[j])

{

flag = false;

break;

}

}

if (flag)

{

tmp.push\_back(a[i]);

}

}

return tmp;

}

template<class T>

void print(miniVector<T> a)

{

for (int i = 0; i < a.size(); i++)

{

cout << a[i] << " ";

}

cout << endl;

}

int main()

{

miniVector<int> a(10);

for (int i = 0; i < 10; i++)

{

cin >> a[i];

}

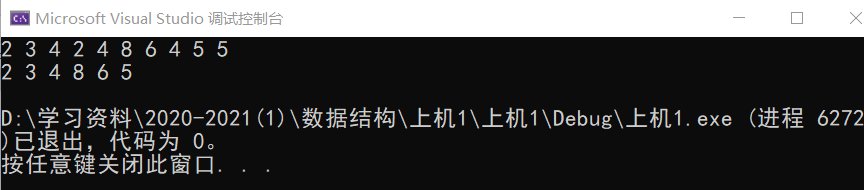
miniVector<int> aa(unique\_NEW(a));

print(aa);

return 0;

}

三、程序运行结果截图



四、讨论分析

1. **运行结果分析**

结果输出为原向量去重后的结果，且原向量中元素相对位置不变。

1. **题目分析**

根据题目描述，要求去重并且避免数据大量移动并保持原始顺序，所以就不能使用删除元素的方法，因为删除元素涉及大量元素的移动，同时也不能使用标志位的方法，因为统计每个元素是否重复需要两重循环，时间复杂度达到，不合适。

考虑到线性表的特性，在尾部追加元素最简单，不需要移动元素，也不需要搜索元素。所以考虑新建一个向量，用来存放去重后的结果。

首先从原向量中选出一个元素，然后遍历一遍新向量，查找没有重复的元素，若没有，则把这个元素追加到新向量的尾部，若有，则继续查看下一个元素。

例如，对于向量（2，5，2，3，5），第一次将2加入新向量中，第二次将5和新向量中的元素（2）比较，没有重复，则把5加入到新向量中，此时新向量为（2，5）。然后取出2与新向量中的元素（2，5）比较，当第一次比较时，发现有重复，那么后面就不用比较了，接着取出下一个元素3，以此类推……示意图如图1所示

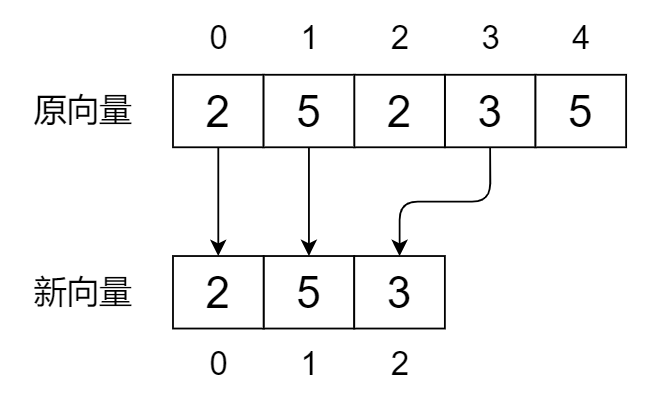


图1 去重示意图

这种方法既避免了元素的大量移动，也充分利用了向量的特性，时间复杂度与空间复杂度较好，而且具有通用性，适用于任何类型的向量。也不会破坏原始数据的相对位置。

1. **程序设计思路**

要实现这一功能，需要一个设计一个函数，参数表为待去重的原向量，返回值为去重后的新向量tmp。同时为了兼顾通用性，需要设计成模板函数，即

template <typename T>

miniVector<T> unique(miniVector<T> a);

在函数内部，首先新建一个向量，用来存放去重后的向量，初始为空向量，然后开始一个for循环，从0到a.size()遍历原向量的每个元素。

在循环内部，先新建一个布尔类型变量flag，初始值为空，用来记录新向量中是否有与现在检查的元素相同的元素。然后再来一个for循环，从0到tmp.size()，逐个判断两个元素a[i]与tmp[j]是否相等，若相等，则flag赋值为false并退出循环。然后判断flag，若flag是true，说明这个元素不是重复元素，则调用push\_back()类函数将a[i]追加到tmp的尾部。

循环执行完毕后，将tmp返回即可。

1. **数学原理说明**

假设待去重的元素个数为n。

在最佳情况下，即待去重向量中所有元素都相等，此时只有第一个元素需要追加到尾部，其它元素只需要比较一次，此时算法的时间复杂度为，此时效率最高。

在最坏情况下，每个元素都需要与新向量中每一个元素比较，而且没有重复元素，所有元素都不同。即第一个向量比较0次，第二个比较1次，第三个比较2次……第n个比较n-1次。此时算法时间复杂度为，取大值为。此时耗费时间最长。

由此可得，待去重元素中重复率越高，则效率越高，反之，效率越低。