**源代码：**

要求1：

#include<iostream>

#include<vector>

using namespace std;

int main()

{

void bubble\_sort(int a[], int n);

int a[] = {25, 45, 65, 78, 74, 11, 24, 72};

int b[sizeof(a)];

memcpy(b, a, sizeof(a));

bubble\_sort(b, 8);

cout<<"原数组：";

for (int i=0; i<8; i++)

cout<<a[i]<<' ';

cout<<endl;

cout<<"修改后：";

for (int i=0; i<8; i++)

cout<<b[i]<<' ';

cout<<endl;

cout<<"原数组：";

for (int i=0; i<8; i++)

cout<<a[i]<<' ';

cout<<endl<<endl;

void bubble\_sort\_vec(vector<int> &v);

vector<int> ve, ve2;

for (int i=0; i<8; i++)

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

ve2 = ve;

bubble\_sort\_vec(ve2);

cout<<"原向量：";

for (int i=0; i<8; i++)

cout<<ve[i]<<' ';

cout<<endl;

cout<<"修改后：";

for (int i=0; i<8; i++)

cout<<ve2[i]<<' ';

cout<<endl;

cout<<"原向量：";

for (int i=0; i<8; i++)

cout<<ve[i]<<' ';

return 0;

}

void bubble\_sort(int a[], int n)

{

int temp;

for (int i=1; i<n; i++)

for (int j=1; j<n-i+1; j++)

if (a[j-1] > a[j]){

temp = a[j];

a[j] = a[j-1];

a[j-1] = temp;

}

}

void bubble\_sort\_vec(vector<int> &v)

{

int temp;

for (int i = 1; i<v.size(); i++)

for(int j=1; j<v.size()-i+1; j++)

if (v[j-1] > v[j]){

temp = v[j];

v[j] = v[j-1];

v[j-1] = temp;

}

}

要求2：

#include <iostream>

#include <vector>

using namespace std;

void cut\_repeat(vector<int> &v)

{

vector<int> temp;

temp.push\_back(v[0]);

for (int i=1; i<v.size(); i++){

bool ju = 1;

for (int j=0; j<temp.size(); j++){

if (v[i]==temp[j])

ju = 0;

}

if (ju)

temp.push\_back(v[i]);

}

v = temp;

}

int main()

{

void cut\_repeat(vector<int> &v);

vector<int> v, v1;

int a[] = {95, 45, 15, 78, 45, 84, 15, 24, 12, 78};

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

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

cut\_repeat(v);

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

cout<<v[i]<<' ';

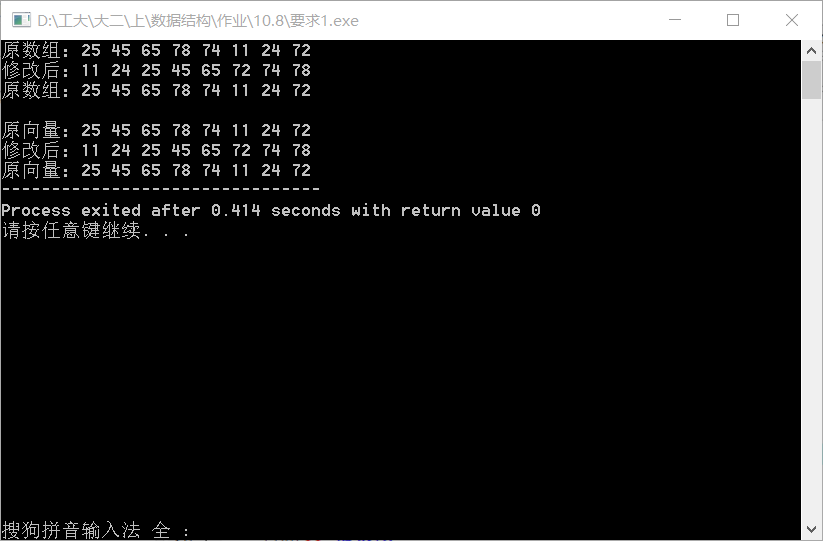
cout<<endl;

return 0;

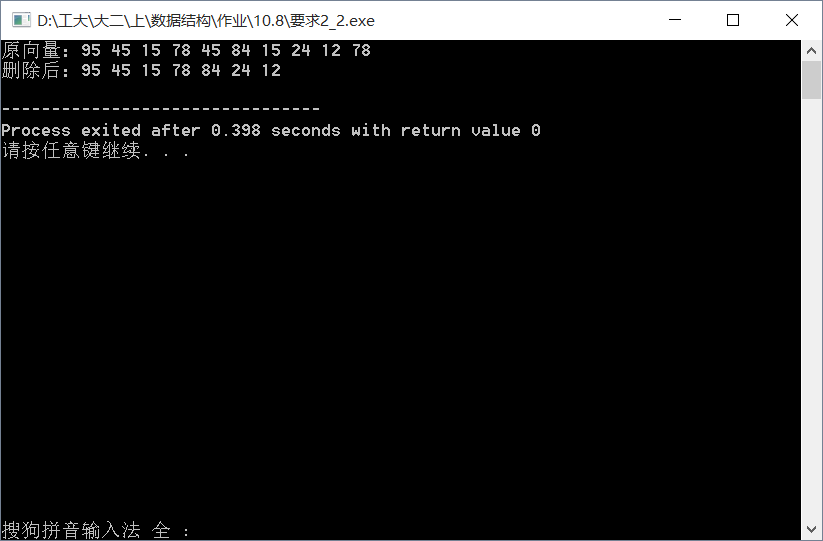
}

**结果截图：**

要求1：



要求2：



**分析：**

要求1要让原来的数据不被破坏，其实只要对原数据做一个备份，让备份去参与冒泡排序就行了，通过空间的占用保证原数据的完整。

要求2原来我是直接基于原向量修改，意味着除了搜寻的复杂度为n2（通过两两比较删除重复项的方式），对原数据的重新排列也会占据复杂度为n的时间。所以我直接在函数中新建了一个向量，将原向量中未重复的数据赋给新建向量，再最后统一复制到原向量中，这意味着没有了删除重复数据之后的重新排列，所以复杂度只有原数据和新建数据比较的n2，速度更快。（当然，直接调用sort函数是最快的选择）