1.反转链表：

# include <iostream>

using namespace std;

struct ListNode {

int val;

struct ListNode \*next;

ListNode(int x) :

val(x), next(NULL) {}

};

class Solution {

public:

ListNode\* ReverList(ListNode\* pHead)

{

//反转指针

ListNode \*pNode = pHead; //当前结点

ListNode \*pPrev = nullptr; //上一结点

ListNode \*pNext = nullptr; //下一结点

ListNode \*pReverseHead = nullptr;

//反转链表

while(pNode != nullptr)

{

pNext = pNode->next; ///建立链表

if (pNext == NULL) //判断pNode是否是最后一个结点

pReverseHead = pNode;

pNode->next = pPrev; //指针翻转

pPrev = pNode;

pNode = pNext;

}

return pReverseHead;

}

};

int main()

{

Solution a;

ListNode a1(1);

ListNode a2(2);

ListNode a3(3);

a1.next = &a2;

a2.next = &a3;

ListNode \*pNode = &a1;

while (pNode!= NULL)

{

cout << pNode->val << endl;

pNode = pNode->next;

}

ListNode \*pReverseHead = a.ReverList(&a1);

while (pReverseHead != NULL)

{

cout << pReverseHead->val << endl;

pReverseHead = pReverseHead->next;

}

getchar();

}

2.冒泡排序

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

//冒泡排序和快排

void bubble\_sort(vector<int> &a)

{

int count = a.size();

for (int i = 0; i< count - 1; i++)

{

for (int j = 0; j< count - i - 1; j++)

{

if (a[j] >a[j + 1])

swap(a[j], a[j + 1]);

}

}

}

int main()

{

//初始化

int n;

int m;

//输入数据

while (cin >> n)

{

vector<int> data;

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

{

cin >> m;

data.push\_back(m);

}

int count = data.size();

//排序

bubble\_sort(data);

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

{

cout << data[i] << ' ';

}

cout << endl;

}

}