**冒泡排序**

**概念：**依次对相邻的两个数进行比较，每一轮下来最大的（最小的）数会自动浮现到最后一位。

**原理：**冒泡排序算法的运作如下：（从后往前）

1、比较相邻的元素。如果第一个比第二个大，就交换他们两个。

2、对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。在这一点，最 后的元素应该会是最大的数。

3、针对所有的元素重复以上的步骤，除了最后一个。

4、持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

**代码：**

#include <iostream>

using namespace std;

template<typename T>

void change\_value(T \*a,T \*b)

{

T temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

template<typename T>

void bubble\_sort(T param[],int n)

{

int i,j;

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

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

if (param[j-1]>param[j])

change\_value(&param[j-1],&param[j]);

}

int main()

{

int arr[] = { 61, 17, 29, 22, 34, 60, 72, 21, 50, 1, 62 };

int len = (int) sizeof(arr) / sizeof(\*arr);

bubble\_sort(arr, len);

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

cout << arr[i] << ' ';

cout << endl;

double arrf[] = { 17.5, 19.1, 0.6, 1.9, 10.5, 12.4, 3.8, 19.7, 1.5, 25.4, 28.6, 4.4, 23.8, 5.4 };

len = (int) sizeof(arrf) / sizeof(\*arrf);

bubble\_sort(arrf, len);

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

cout << arrf[i] << ' ';

cout<<endl;

system("pause");

return 0;

}

**优化代码一：**

//优化：设置一个标识符，如果这一趟发生了交换，则为true，否则为false。如果这一趟没有发生交换，说明排序已经完成。

template<typename T>

void bubble\_sort(T param[],int n)

{

int i;

bool flag=true;

while (flag)

{

flag = false;

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

{

if (param[i-1]>param[i])

{

change\_value(&param[i-1],&param[i]);

flag = true;

}

}

n--;

}

}

**优化代码二：**

//优化二：再做进一步的优化。如果有100个数的数组，仅前面10个无序，后面90个都已//排好序且都大于前面10个数字，

//那么在第一趟遍历后，最后发生交换的位置必定小于10，且这个位置之后的数据必定已经//有序了，记录下这位置，

//第二次只要从数组头部遍历到这个位置就可以了。

template<typename T>

void bubble\_sort(T param[],int n)

{

int i,j;

int flag = n;

while(flag>0)

{

j=flag;

flag=0;

for (i=1;i<j;i++)

{

if (param[i-1]>param[i])

{

change\_value(&param[i-1],&param[i]);

flag=i;

}

}

}

}

**算法分析：**

**（1）算法的最好时间复杂度**

    　若文件的初始状态是正序的，一趟扫描即可完成排序。所需的关键字比较次数C和记录移动次数M均达到最小值：

        Cmin=n-1

        Mmin=0。

    　冒泡排序最好的时间复杂度为O(n)。

**（2）算法的最坏时间复杂度**

    　若初始文件是反序的，需要进行n-1趟排序。每趟排序要进行n-i次关键字的比较(1≤i≤n-1)，且每次比较都必须移动记录三次来达到交换记录位置。在这种情况下，比较和移动次数均达到最大值：

        Cmax=n(n-1)/2=O(n2)

        Mmax=3n(n-1)/2=O(n2)

冒泡排序的最坏时间复杂度为O(n2)。

**（3）算法的平均时间复杂度为O(n2)**

    　虽然冒泡排序不一定要进行n-1趟，但由于它的记录移动次数较多，故平均时间性能比直接插入排序要差得多。

**（4）算法稳定性**

冒泡排序是就地排序，且它是稳定的。

**（5）算法评价**

冒泡排序毕竟是一种效率低下的排序方法，在数据规模很小时，可以采用。数据规模比较大时，最好用其它排序方法。