**直接插入排序**

**概念：**把待排序的记录按其关键码值的大小逐个插入已排好序的有序序列中；通俗的说法就是每次从无序表中取出第一个元素，把它插入到有序表的合适位置，使有序表仍然有序。

**原理：**直接排序算法的思路：

1、设置监视哨r[0]，将待插入记录赋值给r[0]；

2、设置开始查找的位置j；

3、在数组中进行搜索，搜索中将第j个记录后移，直至r[0].key>=r[j].key为止；

4、将r[0]插入r[j+1]的位置上。

通俗的描述为：

1、将原序列分成有序区和无序区。a[0...i-1]为有序区，a[i...n]为无序区。（i从1开始）

2、从无序区中取出第一个元素，即a[i]，在有序区序列中从后向前扫描。

3、如果有序元素大于a[i]，将有序元素后移到下一位置。

4、重复步骤3，直到找到小于或者等于a[i]的有序元素，将a[i]插入到该有序元素的下 一个位置中。

5、重复2~4，直到无序区最后。

**代码：**

#include "stdafx.h"

#include <iostream>

using namespace std;

template<typename T>

void straight\_insert\_sort(T a[],int n)

{

int j;

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

{

T tmp = a[i];

for (j=i-1;j>=0 && a[j]>tmp;j--)

{

a[j+1] = a[j];

}

a[j+1] = tmp;

}

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

double a[] = {98.5,76.6,109,34,67,190,80,12,14,89,1};

int n = (int)sizeof(a)/sizeof(a[0]);

straight\_insert\_sort(a,n);

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

cout<<a[f]<<" ";

cout<<endl;

system("pause");

return 0;

}

**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，需要进行的比较操作需（n-1）次。后移赋值操作为0次。　直接插入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，那么此时需要进行的比较共有n(n-1)/2次。后移赋值操作是比较操作的次数加上n-1次。

直接插入排序的最坏时间复杂度为O(n^2)。

**（3）算法的平均时间复杂度为O(n^2)**

**（4）算法稳定性**

直接插入是稳定的排序。

**（5）算法评价**

算法适用于少量的排序，最坏时间复杂度是O（N^2），空间复杂度是O（1）。