测试头文件内容

#include<iostream>

#include<ctime>

#include<cassert>

using namespace std;

// 生成有n个元素的随机数组，每个元素的随机范围为[rangeL,rangeR]

int\* generateRandomArray(int n,int rangeL,int rangeR){

assert( rangeL <= rangeR );

int \*arr = new int[n];

srand(time(NULL));

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

arr[i] = rand() % (rangeR - rangeL + 1) + rangeL;

}

return arr;

}

// 打印数组

template <typename T>

void printArr(T arr[],int n){

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

cout<<arr[i]<<" ";

cout<<endl;

}

// 排序正确性

template<typename T>

bool isSorted(T arr[] , int n){

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

if(arr[i] > arr[i+1])

return false;

return true;

}

// 测试数组

template <typename T>

void testArr(string sortName , void(\*sort)(T arr[] , int) , T arr[], int n){

clock\_t startTime = clock();

sort(arr , n);

clock\_t endTime = clock();

assert(isSorted(arr,n));

cout<<sortName<<": "<< double(endTime - startTime) / CLOCKS\_PER\_SEC <<"s"<<endl;

}

**选择排序**

template <typename T>

void selectionSort(T arr[] , int n){

int temp = 0;

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

int minIndex = i;

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

if(arr[minIndex] > arr[j])

minIndex = j;

}

swap(arr[minIndex] , arr[i]);

}

}

**插入排序**

从第二个元素开始，因为也相对来说第一个元素已经排好位置了。

template<typename T>

void insertionSort(T arr[] , int n){

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

for( int j = i ; j > 0 && arr[j] < arr[j-1] ; j --){

swap(arr[j],arr[j-1]);

}

}

}

由于插入排序会提前终止排序过程，所以排序时间相对较短。

未经过优化的插入排序运行时间超过选择排序，原因是插入排序在排序过程中，不但有比较的时间，还有移动交换元素的时间。