#include <stdio.h> // Quick Sort 1

int a[101],n; //定义全局变量，这两个变量需要在子函数中使用

void quicksort(int left,int right)

{

int i,j,t,temp;

if(left>right) return;

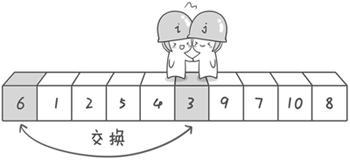
temp=a[left]; // temp中存的就是基准数

i=left;

j=right;

while(i!=j)

{

 while(a[j]>=temp && i<j) // 顺序很重要，要先从右边开始找

j--;

while(a[i]<=temp && i<j) // 再找左边的

i++;

if(i<j) // 交换两个数在数组中的位置

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

a[left]=a[i]; // 最终将基准数归位

a[i]=temp;

quicksort(left,i-1); // 继续处理左边的，这里是一个递归的过程

quicksort(i+1,right); // 继续处理右边的 ，这里是一个递归的过程

}

int main()

{

int i,j,n;

printf("输入需要处理的元素个数:");//读入数据

scanf("%d",&n);

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

{

printf("输入第%d个元素（共%d个元素):",i,n);

scanf("%d",&a[i-1]);

}

quicksort(1,n); // 快速排序调用

for(i=1;i<=n;i++) // 输出排序后的结果

printf("%d ",a[i]);

printf("\n");

system("pause");

return 0;

}

#include <iostream> // Quick Sort 2

using namespace std;

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5** | **2** | **9** | **4** | **7** | **6** | **1** | **3** | **8** |
| **3** | **2** | **9** | **4** | **7** | **6** | **1** | **5** | **8** |
| **3** | **2** | **1** | **4** | **7** | **6** | **9** | **5** | **8** |

int Partition(int r[],int low,int high) // 划分函数和基准元素交换

{

int i=low,j=high,pivot=r[low]; // 基准元素

while(i<j)

{

while(i<j && r[j]>pivot) // 向左扫描

j--;

if(i<j)

swap(r[i++],r[j]); // r[i]和r[j]交换后i+1右移一位

while(i<j && r[i]<=pivot) // 向右扫描

i++;

if(i<j)

swap(r[i],r[j--]); // r[i]和r[j]交换 后j-1左移一位

}

return i; // 返回最终划分完成后基准元素所在的位置

}

void QuickSort(int R[],int low,int high) // 实现快排算法

{

int mid;

if(low<high)

{

mid=Partition(R,low,high); // 基准位置

QuickSort(R,low,mid-1); // 左区间递归快排

QuickSort(R,mid+1,high); // 右区间递归快排

}

}

int main()

{

int arr[] = {}; // 从小到大快速排序

5,2,9,4,7,6,1,3,8

int n = sizeof(arr) / sizeof(int);

QuickSort(arr,0,n-1);

cout<<"排序后的序列为："<<endl;

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

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

cout<<endl;

system("pause");

return 0;

}

分类：内部比较排序；数据结构：数组；稳定性：不稳定；

最差时间复杂度：每次选取的基准都是最大（或最小）的元素，导致每次只划分出了一个分区，需要进行n-1次划分才能结束递归，时间复杂度为O(n^2)

最优时间复杂度：每次选取的基准都是中位数，这样每次都均匀的划分出两个分区，只需要logn次划分就能结束递归，时间复杂度为O(nlogn)

平均时间复杂度：O(nlogn)

所需辅助空间：主要是递归造成的栈空间的使用(用来保存left和right等局部变量)，取决于递归树的深度，一般为O(logn)，最差为O(n)

#include <stdio.h> // Quick Sort 3

#include <stdlib.h>

void Swap(int A[], int i, int j)

{

int temp = A[i];

A[i] = A[j];

A[j] = temp;

}

int Partition(int A[], int left, int right) // 划分函数

{

int pivot = A[right]; // 这里每次都选择最后一个元素作为基准

//int pivot = A[(left+right)/2]; // 也可以每次都选择中间的元素作为基准

int tail = left - 1; // tail为小于基准的子数组最后一个元素的索引

for (int i = left; i < right; i++) // 遍历基准以外的其他元素

{

if (A[i] <= pivot) // 把小于等于基准的元素放到前一个子数组末尾

Swap(A, ++tail, i);

}

Swap(A, tail + 1, right); // 最后把基准放到前一个子数组的后边，剩下的子数组即是大于基准的子数组

// 该操作很有可能把后面元素的稳定性打乱，所以快速排序是不稳定的排序算法

return tail + 1; // 返回基准的索引

8是基准

0 1 2 3 4 5 6 7 8 i tail

5 2 9 4 11 6 1 3 8 0 -1

5 2 9 4 11 6 1 3 8 0 0

5 2 9 4 11 6 1 3 8 1 1

5 2 9 4 11 6 1 3 8 2 1

5 2 **4 9** 11 6 1 3 8 3 2

5 2 4 9 11 6 1 3 8 4 2

5 2 4 **6** 11 **9** 1 3 8 5 3

5 2 4 6 **1** 9 **11** 3 8 6 4

5 2 4 6 1 **3** 11 **9** 8 7 5

5 2 4 6 1 3 **8** 9 **11** 8 6

}

void QuickSort(int A[], int left, int right)

{

if (left >= right)

return;

int pivot\_index = Partition(A, left, right); // 基准的索引

QuickSort(A, left, pivot\_index - 1);

QuickSort(A, pivot\_index + 1, right);

}

int main()

{

int A[] = {5,2,9,4,11,6,1,3,8}; // 从小到大快速排序

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

QuickSort(A, 0, n-1);

printf("快速排序结果：");

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

printf("%d ", A[i]);

printf("\n");

system("pause");

return 0;

}快速排序（Quick sort）是对冒泡排序的一种改进。快速排序由C. A. R. Hoare在1962年提出。它的基本思想是：通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。