4.C语言快速排序算法及代码

快速排序是对冒泡法排序的一种改进。

快速排序算法 的基本思想是：将所要进行排序的数分为左右两个部分，其中一部分的所有数据都比另外一 部分的数据小，然后将所分得的两部分数据进行同样的划分，重复执行以上的划分操作，直到所有要进行排序的数据变为有序为止。

可能仅根据基本思想对快速排序的认识并不深，接下来以对n个无序数列A[0], A[1]…, A[n-1]采用快速排序方法进行升序排列为例进行讲解。

(1)定义两个变量low和high，将low、high分别设置为要进行排序的序列的起始元素和最后一个元素的下标。第一次，low和high的取值分别为0和n-1，接下来的每次取值由划分得到的序列起始元素和最后一个元素的下标来决定。

(2)定义一个变量key，接下来以key的取值为基准将数组A划分为左右两个部分，通 常，key值为要进行排序序列的第一个元素值。第一次的取值为A[0]，以后每次取值由要划 分序列的起始元素决定。

(3)从high所指向的数组元素开始向左扫描，扫描的同时将下标为high的数组元素依次与划分基准值key进行比较操作，直到high不大于low或找到第一个小于基准值key的数组元素，然后将该值赋值给low所指向的数组元素，同时将low右移一个位置。

(4)如果low依然小于high，那么由low所指向的数组元素开始向右扫描，扫描的同时将下标为low的数组元素值依次与划分的基准值key进行比较操作，直到low不小于high或找到第一个大于基准值key的数组元素，然后将该值赋给high所指向的数组元素，同时将high左移一个位置。

(5)重复步骤(3) (4)，直到low的植不小于high为止，这时成功划分后得到的左右两部分分别为A[low……pos-1]和A[pos+1……high]，其中，pos下标所对应的数组元素的值就是进行划分的基准值key，所以在划分结束时还要将下标为pos的数组元素赋值 为 key。

(6)将划分得到的左右两部分A[low……pos-1]和A[pos+1……high]继续采用以上操作步骤进行划分，直到得到有序序列为止。

为了能够加深读者的理解，接下来通过一段代码来了解快速排序的具体实现方法。

```
01.   #include <stdio.h>
02.   #include <stdlib.h>
03.   #define N 6
04.
05.   int partition(int arr[], int low, int high){
```

```
06.       int key;
07.       key = arr[low];
08.       while(low<high){
09.           while(low <high && arr[high]>= key )
10.               high--;
11.           if(low<high)
12.               arr[low++] = arr[high];
13.           while( low<high && arr[low]<=key )
14.               low++;
15.           if(low<high)
16.               arr[high--] = arr[low];
17.       }
18.       arr[low] = key;
19.
20.       return low;
21.   }
22.
23.   void quick_sort(int arr[], int start, int end){
24.       int pos;
25.       if (start<end){
26.           pos = partition(arr, start, end);
27.           quick_sort(arr,start,pos-1);
28.           quick_sort(arr,pos+1,end);
29.       }
30.
31.       return;
32.   }
33.
34.   int main(void){
35.       int i;
36.       int arr[N]={32,12,7, 78, 23,45};
37.
38.       printf("排序前 \n");
39.       for(i=0;i<N;i++)
40.           printf("%d\t",arr[i]);
41.       quick_sort(arr,0,N-1);
42.
43.       printf("\n 排序后 \n");
44.       for(i=0; i<N; i++)
45.           printf("%d\t", arr[i]);
46.       printf ("\n");
```

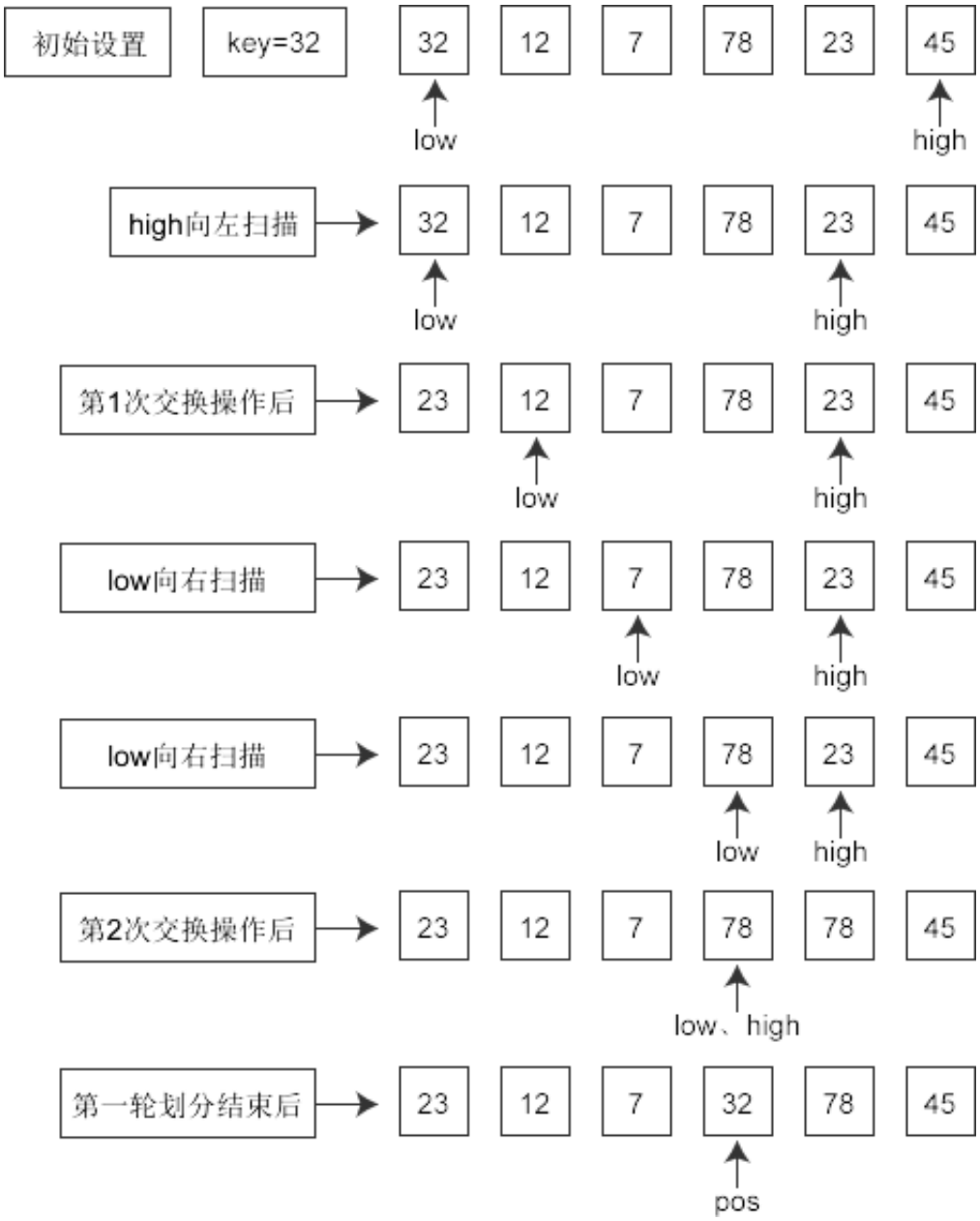
```
47.
48.        system("pause");
49.        return 0;
50.    }
```

运行结果：

```
排序前
32     12     7     78     23     45
排序后
7     12     23     32     45     78
```

在上面的代码中，根据前面介绍的步骤一步步实现了快速排序算法。接下来通过示意图来演示第一次划分操作。

在第一次划分操作中，先进行初始设置，key的值是进行划分的基准，其值为要划分数 组的第一个元素值，在上面的排序序列中为第一个元素值32，同时将low设置为要排序数组中第一个元素的下标，第一次排序操作时其值为0，将high设置为要排序序列最后一个 元素的下标，在上面的排序序列中其第一次取值为5。先将下标为high的数组元素与key进行比较，由于该元素值大于key，因此high向左移动一个位置继续扫描。由于接下来的值为 23，小于key的值，因此将23赋值给下标为low所指向的数组元素。接下来将low右移一 个位置，将low所指向的数组元素的值与key进行比较，由干接下来的12、7都小于key， 因此low继续右移扫描，直至下标low所指向的数组元素的值为78即大于key为止，将78赋值给下标为high所指向的数组元素，同时将high左移一个位置。接下来由于low不再小于high，划分结束。需要注意的是，在进行划分的过程中，都是将扫描的值与key的值进行对比，如果小于key，那么将该值赋值给数组中的另外一个元素，而该元素的值并没有改变。从图中可以看出这一点，所以需要在划分的最后将作为划分基准的key值赋值给下标为 pos的数组元素，这个元素不再参与接下来的划分操作。



第一次划分操作

第一轮划分结束后，得到了左右两部分序列A[0]、A[1]、A[2]和A[4]、A[5]，继续进 行划分，即对每轮划分后得到的两部分序列继续划分，直至得到有序序列为止。