/\*\*

\* QuickSort.c 快速排序算法

\*

\* @author Darbuly 970073804@qq.com

\* @copyright 2019-2020 DBL

\*

\*/

#include <stdio.h>

void swap(int k[],int low,int high)

{

int temp;

temp = k[low];

k[low] = k[high];

k[high] = temp;

}

int Partition(int k[],int low,int high)

{

int point;

point = k[low];

while(low<high)

{

while(low<high && k[high]>=point)

{

high--;

}

swap(k,low,high);

while(low<high && k[low]<=point)

{

low++;

}

swap(k,low,high);

}

return low;

}

void QSort(int k[],int low,int high)

{

int point;

if(low<high)

{

point = Partition(k,low,high);

QSort(k,low,point-1);

QSort(k,point+1,high);

}

}

void QuickSort(int k[],int n)

{

QSort(k,0,n-1);

}

int main()

{

int i,a[10] = {5,2,6,0,3,9,1,7,4,8};

QuickSort(a,10);

printf("Sort by QuickSort\n");

for(i=0;i<10;i++)

{

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

}

printf("\n\n");

return 0;

}

/\*\*

\* QuickSort.c 快速排序算法 优化选取基准点

\*

\* @author Darbuly 970073804@qq.com

\* @copyright 2019-2020 DBL

\*

\*/

#include <stdio.h>

void swap(int k[],int low,int high)

{

int temp;

temp = k[low];

k[low] = k[high];

k[high] = temp;

}

int Partition(int k[],int low,int high)

{

int point;

//优化选取基准点

int m = low + (high-low)/2;

if(k[low]>k[high])

{

swap(k,low,high);

}

if(k[m]>k[high])

{

swap(k,m,high);

}

if(k[m]>k[low])

{

swap(k,m,low);

}

point = k[low];

while(low<high)

{

while(low<high && k[high]>=point)

{

high--;

}

swap(k,low,high);

while(low<high && k[low]<=point)

{

low++;

}

swap(k,low,high);

}

return low;

}

void QSort(int k[],int low,int high)

{

int point;

if(low<high)

{

point = Partition(k,low,high);

QSort(k,low,point-1);

QSort(k,point+1,high);

}

}

void QuickSort(int k[],int n)

{

QSort(k,0,n-1);

}

int main()

{

int i,a[10] = {5,2,6,0,3,9,1,7,4,8};

QuickSort(a,10);

printf("Sort by QuickSort\n");

for(i=0;i<10;i++)

{

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

}

printf("\n\n");

return 0;

}

/\*\*

\* QuickSort.c 快速排序算法 优化不必要的交换

\*

\* @author Darbuly 970073804@qq.com

\* @copyright 2019-2020 DBL

\*

\*/

#include <stdio.h>

void swap(int k[],int low,int high)

{

int temp;

temp = k[low];

k[low] = k[high];

k[high] = temp;

}

int Partition(int k[],int low,int high)

{

int point;

point = k[low];

while(low<high)

{

while(low<high && k[high]>=point)

{

high--;

}

//swap(k,low,high);

k[low] = k[high];//优化不必要的交换

while(low<high && k[low]<=point)

{

low++;

}

//swap(k,low,high);

k[high] = k[low];//优化不必要的交换

}

k[low] = point;

return low;

}

void QSort(int k[],int low,int high)

{

int point;

if(low<high)

{

point = Partition(k,low,high);

QSort(k,low,point-1);

QSort(k,point+1,high);

}

}

void QuickSort(int k[],int n)

{

QSort(k,0,n-1);

}

int main()

{

int i,a[10] = {5,2,6,0,3,9,1,7,4,8};

QuickSort(a,10);

printf("Sort by QuickSort\n");

for(i=0;i<10;i++)

{

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

}

printf("\n\n");

return 0;

}

/\*\*

\* QuickSort.c 快速排序算法 优化小数组的解决方案

\*

\* @author Darbuly 970073804@qq.com

\* @copyright 2019-2020 DBL

\*

\*/

#include <stdio.h>

#define MAX\_LENGTH\_INSERT\_SORT 7

void ISort(int k[],int n)

{

int i,j,temp;

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

{

if(k[i]<k[i-1])

{

temp = k[i];

for(j=i-1;k[j]>temp;j--)

{

k[j+1] = k[j];

}

k[j+1] = temp;

}

}

}

void InsertSort(int k[],int low,int high)

{

ISort(k+low,high-low+1);

}

void swap(int k[],int low,int high)

{

int temp;

temp = k[low];

k[low] = k[high];

k[high] = temp;

}

int Partition(int k[],int low,int high)

{

int point;

point = k[low];

while(low<high)

{

while(low<high && k[high]>=point)

{

high--;

}

swap(k,low,high);

while(low<high && k[low]<=point)

{

low++;

}

swap(k,low,high);

}

return low;

}

void QSort(int k[],int low,int high)

{

int point;

if(high-low>MAX\_LENGTH\_INSERT\_SORT)

{

point = Partition(k,low,high);

QSort(k,low,point-1);

QSort(k,point+1,high);

}

else

{

InsertSort(k,low,high);

}

}

void QuickSort(int k[],int n)

{

QSort(k,0,n-1);

}

int main()

{

int i,a[10] = {5,2,6,0,3,9,1,7,4,8};

QuickSort(a,10);

printf("Sort by QuickSort\n");

for(i=0;i<10;i++)

{

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

}

printf("\n\n");

return 0;

}

/\*\*

\* QuickSort.c 快速排序算法 优化递归操作，变成尾递归

\*

\* @author Darbuly 970073804@qq.com

\* @copyright 2019-2020 DBL

\*

\*/

#include <stdio.h>

void swap(int k[],int low,int high)

{

int temp;

temp = k[low];

k[low] = k[high];

k[high] = temp;

}

int Partition(int k[],int low,int high)

{

int point;

point = k[low];

while(low<high)

{

while(low<high && k[high]>=point)

{

high--;

}

swap(k,low,high);

while(low<high && k[low]<=point)

{

low++;

}

swap(k,low,high);

}

return low;

}

void QSort(int k[],int low,int high)

{

int point;

if(low<high)

{

while(low<high)

{

point = Partition(k,low,high);

QSort(k,low,point-1);

low = point+1;

}

}

}

void QuickSort(int k[],int n)

{

QSort(k,0,n-1);

}

int main()

{

int i,a[10] = {5,2,6,0,3,9,1,7,4,8};

QuickSort(a,10);

printf("Sort by QuickSort\n");

for(i=0;i<10;i++)

{

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

}

printf("\n\n");

return 0;

}