<QuickSort 소스코드>

#include <stdio.h>

typedef struct vec

{

int x;

}vec;

void Swap(vec \*arr, int idx1, int idx2)

{

int temp = arr[idx1].x;

arr[idx1].x = arr[idx2].x;

arr[idx2].x = temp;

}

int Partition(vec \*arr, int left, int right)

{

int pivot = arr[left].x; // 피벗의 위치는 가장 왼쪽!

int low = left+1;

int high = right;

int i;

while(low <= high) // 교차되지 않을 때까지 반복

{

while(pivot > arr[low].x)

low++;

while(pivot < arr[high].x)

high--;

/\*

while(pivot >= arr[low] && low <= right)

low++;

while(pivot <= arr[high] && high >= (left+1))

high--;

\*/

if(low <= high) // 교차되지 않은 상태라면 Swap 실행

Swap(arr, low, high); // low와 high가 가리키는 대상 교환

printf("p1: ");

for(i=left;i<=right;++i) printf("%d ",arr[i].x);

printf("\n");

}

Swap(arr, left, high); // 피벗과 high가 가리키는 대상 교환

printf("p2: ");

for(i=left;i<=right;++i) printf("%d ",arr[i].x);

printf("\n");

return high; // 옮겨진 피벗의 위치 정보 반환

}

void QuickSort(vec \*arr, int left, int right)

{

int i;

if(left <= right)

{

int pivot = Partition(arr, left, right); // 둘로 나눠서

for(i=left;i<=right;++i) printf("%d ",arr[i].x);

printf("\n");

QuickSort(arr, left, pivot-1); // 왼쪽 영역을 정렬

QuickSort(arr, pivot+1, right); // 오른쪽 영역을 정렬

}

}

int main(void)

{

vec arr[7] = {

{3}, {2}, {4}, {1}, {7}, {6}, {5}

};

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

int i;

QuickSort(arr, 0, sizeof(arr)/sizeof(int)-1);

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

printf("%d ", arr[i].x);

printf("\n");

return 0;

}

<결과>

