[417p. 퀵 정렬 함수 quick\_sort]

void quick\_sort(int a[], int left, int right) {

int pivot, i, j, tmp;

if (left < right) {

i = left; j = right + 1; pivot = a[left];

while (i < j) {

do i++; while ((a[i] < pivot) && (i < right));

do j++; while ((a[j] > pivot) && (j < left));

if (i < j) {

tmp = a[i]; a[i] = a[j]; a[j] = tmp;

}

}

if (j != left) {

tmp = a[j];

a[j] = a[left];

a[left] = tmp;

}

quick\_sort(a, left, j - 1);

quick\_sort(a, j + 1, right);

}

}

/\*

240111 6번문제 퀵 정렬

10-11주차 pdf 참조

평균적으로 수행속도가 빠른 방법으로 사용됨

기본 정렬

제어 값을 중심으로 두개의 데이터 집합으로 분할한다.

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

void quick\_sort(int a[], int left, int right) {

int pivot, i, j, tmp;

if (left < right) {

i = left; j = right + 1; pivot = a[left];

while (i < j) {

do i++; while ((a[i] < pivot) && (i < right));

do j--; while ((a[j] > pivot) && (j < left));

if (i < j) {

tmp = a[i]; a[i] = a[j]; a[j] = tmp;

}

}

if (j != left) {

tmp = a[j];

a[j] = a[left];

a[left] = tmp;

}

quick\_sort(a, left, j - 1);

quick\_sort(a, j + 1, right);

}

}

//////////////////////////////////////////////////////////////////////////////////////////////////

typedef struct employee {

long eid;

char name[20];

int escore;

char dname[5];

}edatatype;

void quick\_sorta(edatatype a[], int left, int right);

void quick\_sortd(edatatype a[], int left, int right);

void quick\_sorta(edatatype a[], int left, int right) {

long pivot;

int i, j;

edatatype tmp;

if (left < right) {

i = left; j = right + 1; pivot = a[left].eid;

while (i < j) {

do i++; while ((a[i].eid <= pivot) && (i < right));

do j--; while ((a[j].eid >= pivot) && (j > left));

if (i < j) {

tmp = a[i];

a[i] = a[j];

a[j] = tmp;

}

}

if (j != left) {

tmp = a[j];

a[j] = a[left];

a[left] = tmp;

}

quick\_sorta(a, left, j - 1);

quick\_sorta(a, j + 1, right);

}

}

void quick\_sortd(edatatype a[], int left, int right)

{

int pivot;

int i, j;

edatatype tmp;

if (left < right) {

i = left; j = right + 1;

pivot = a[left].escore;

while (i < j) {

do i++; while ((a[i].escore >= pivot) && (i < right));

do j--; while ((a[j].escore <= pivot) && (j > left));

if (i < j) {

tmp = a[i];

a[i] = a[j];

a[j] = tmp;

}

}

if (j != left) {

tmp = a[j];

a[j] = a[left];

a[left] = tmp;

}

quick\_sortd(a, left, j - 1);

quick\_sortd(a, j + 1, right);

}

}

void main(int argc, char \*argv[])

{

edatatype edb[50];

FILE \*infile, \*out;

int i = 0, j;

char choice;

char dcode[5];

if (argc != 3) {

printf("실행 인수가 잘못되었습니다. \n");

exit(1);

}

if ((infile = fopen(argv[1], "r")) == NULL) {

printf("입력파일을 열 수 없습니다. \n");

exit(1);

}

if ((out = fopen(argv[2], "w")) == NULL) {

printf("결과 파일을 열 수 없습니다 \n");

exit(1);

}

while (fscanf(infile, "%ld %s %d %s", &(edb[i].eid), edb[i].name, &(edb[i].escore), edb[i].dname) != EOF)

i++;

//1. id의 오름차순으로 파일에 저장

quick\_sorta(edb, 0, i - 1);

printf("\n\nSorted Data ... \n");

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

fprintf(out, "%ld\t\%s\t%d\t%s\n", (edb[j].eid), edb[j].name, (edb[j].escore), edb[j].dname);

//2. dcode를 읽어 그 부서에 해당하는 사원정보 출력하기

printf("Sales Department : S110\n");

printf("Research Department: R250\n");

printf("Management Department: M210\n");

printf("\nEnter the Department code: ");

scanf("%s", dcode);

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

if (strcmp(dcode, edb[j].dname) == 0)

printf("%ld\t%s\t%d\n", (edb[j].eid), edb[j].name, (edb[j].escore));

//3. 영어 성적이 큰 값부터 정보 출력하기

printf("\n \nFrom the hightest English score to the lowest...\n");

quick\_sortd(edb, 0, i - 1);

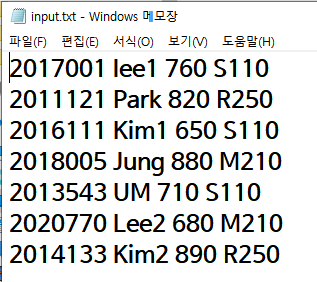
for (j = 0; j < i; j++)

printf("%ld\t%s\t%d\t%s\n", (edb[j].eid), edb[j].name, (edb[j].escore), edb[j].dname);

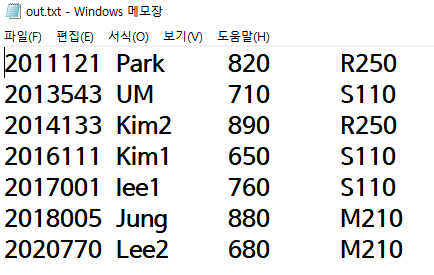
}



[INPUT.TXT]



[OUT.TXT]



/\*

240112 3번문제 합병 정렬 10-15

13주차 pdf 참조

정렬된 두 데이터 집합을 하나의 정렬된 데이터집합으로 합병

a1, a2, a의 현재 위치를 나타내는 인덱스를 i,j,k로 지정(이때 초기화 진행) => i=j=k=0

a1[i] <= a2[j]이면 a[k] = a1[i], i++, k++

a1[i] > a2[j]이면 a[k] = a2[j], j++, k++

어느 한 리스트를 다 처리하면 나머지 리스트의 데이터를 복사한다.

a1의 데이터의 수 n1, a2의 데이터의 수를 n2라고 지정

while (j <= n2) a[k]=a2[j], k++, j++

while (i <= n2) a[k]=a1[i], k++, i++

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

void print\_list(int \*list, int n, char \*mesg);

void quick\_sort(int a[], int left, int right);

int two\_one(int \*a1, int \*a2, int \*a, int n1, int n2);

void main()

{

int list1[] = { 32, 15, 20, 50, 40, 10, 25, 30 };

int list2[] = { 25, 50, 15, 17, 20, 10, 38, 13, 35 };

int list[100];

int n1, n2, n;

char sorted;

n1 = sizeof(list1) / sizeof(int);

n2 = sizeof(list2) / sizeof(int);

print\_list(list1, n1, "데이터 리스트 1: \n");

print\_list(list2, n2, "데이터 리스트 2: \n");

printf("\n 준비된 데이터 리스트1이 정렬되어 있는가? (Y/N)");

sorted = getche();

if (sorted == 'n') quick\_sort(list1, 0, n1 - 1);

printf("\n 준비된 데이터 리스트2가 정렬되어 있는가? (Y/N");

sorted = getche();

if (sorted == 'n') quick\_sort(list2, 0, n2 - 1);

printf("\n\n");

n = two\_one(list1, list2, list, n1, n2);

print\_list(list, n, "합병된 데이터 리스트: \n");

}

void print\_list(int \*list, int n, char \*mesg)

{

int i;

printf(mesg);

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

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

if ((i % 10) == 9) printf("\n");

}

printf("\n\n");

}

int two\_one(int\* a1, int\* a2, int\* a, int n1, int n2)

{

int i = 0, j = 0, k = 0;

while (i < n1 && j < n2) {

if (a1[i] <= a2[j])

a[k++] = a1[i++];

else if (a1[i] > a2[j])

a[k++] = a2[j++];

}

if (i == n1)

while (j < n2) a[k++] = a2[j++];

else

while (i < n1) a[k++] = a1[i++];

return k;

}

void quick\_sort(int a[], int left, int right) {

int pivot, i, j, tmp;

if (left < right) {

i = left; j = right + 1; pivot = a[left];

while (i < j) {

do i++; while ((a[i] < pivot) && (i < right));

do j--; while ((a[j] > pivot) && (j > left));

if (i < j) {

tmp = a[i];

a[i] = a[j];

a[j] = tmp;

}

}

if (j != left) {

tmp = a[j];

a[j] = a[left];

a[left] = tmp;

}

quick\_sort(a, left, j - 1);

quick\_sort(a, j + 1, right);

}

}

