강의명: 자료구조

숙제 번호: 9

숙제 제목: Dictionary file sort(사전 파일 정렬)

학생 이름: 김성현

학번: 201910783

1. Dictionary file sort(사전 파일 정렬)

1.1

// ======================================================================

#include <stdio.h>

#include <stdlib.h>

#include "sort.h"

// ======================================================================

typedef struct {

int key;

char word[WORD\_SIZE];

} element;

element dict[DICT\_SIZE];

// ======================================================================

/\*element sorted[DICT\_SIZE];

void merge(element \*list, int left, int mid, int right)

{

int i, j, k, l;

i = left; j = mid + 1; k = left;

while(i<mid && j<=right) {

if(list[i]->key <= list[j]->key)

sorted[k++] = list[i++];

else

sorted[k++] = list[j++];

}

if(i>mid)

for(l=j; l<=right; l++)

sorted[k++] = list[l];

else

for(l=i; l<=mid; l++)

sorted[k++] = list[l];

for(l=left; l<=right; l++)

list[l] = sorted[l];

}

\*/

// ======================================================================

/\*

void merge\_sort(element \*list, int left, int right)

{

int mid;

if(left<right) {

mid = (left+right)/2;

merge\_sort(list, left, mid);

merge\_sort(list,mid+1, right);

merge(list, left, mid, right);

}

}

\*/

// ======================================================================

/\*void insertion\_sort(element \*list, int n) {

int i, j;

element key[1];

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

key[0] = list[i];

for(j = i-1; j>=0 && list[j].key>key[0].key; j--)

list[j+1] = list[j];

list[j+1] = key[0];

}

}

\*/

// ======================================================================

void sort(void)

{

FILE \* fp;

int ret;

fp = fopen(DICT\_IN, "rt");

if(fp==NULL) {

puts("fail to open file!");

exit(0);

}

int i = 0;

while(1){

ret = fscanf(fp, "%d %s", &dict[i].key, dict[i].word);

if(ret == EOF)

break;

i++;

}

fclose(fp);

//rhe shortest sorting way on the table is "merge" & "heap", but it's a little difficult.

//so, Let's do with simple sorting ways.

/\*element test[DICT\_SIZE];

for(int z = 0; z < DICT\_SIZE; z++) {

test[z] = dict[z];

}

\*/

//i) insertion\_sort

int l, j;

element tmp[1];

for(l = 1; l <= i; l++) {

tmp[0] = dict[i];

for(j = l - 1; (j>=0) && (dict[j].key>tmp[0].key); j--)

dict[j+1] = dict[j];

dict[j+1] = tmp[0];

}

//ii) merge\_sort

/\*element tmp[i+1];

for(int v = 0; v < i; v++) {

tmp[v] = dict[v];

}

merge\_sort(tmp, 0, i);

\*/

//insertion\_sort(dict, i);

FILE \*fin = fopen(DICT\_OUT, "wt");

int a = 0;

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

fprintf(fin, "%d %s\n", dict[a].key, dict[a].word);

}

fclose(fin);

}

// ======================================================================

\*이렇게 프로그램한 이유:

먼저 DICT\_IN.txt에 있는 것들을 받아오기 위해 FILE 포인터와 fscanf함수를 통해 dict구조체 배열에 저장을 시켜 사용할 준비를 하였습니다. 그 다음 본격적으로 정렬을 시도하였는데, 결론적으로는 모두 실패하였지만 간단하게 설명해드리자면 우선 가장 간단한 정렬 방법이자 하나의 함수로 구현이 가능한 삽입정렬을 시도하여 실험삼아 하려고 하였습니다. 그렇기에 for문 중첩을 통해 중간에 삽입하는 형식의 삽입 정렬을 시도하였으나 실패하였고, 그 다음 방법으로는 합병정렬을 시도하였습니다. 지금까지 배운 정렬 방법 중 합병정렬과 히프정렬이 가장 종합적으로 따졌을 때 가장 적은 시간이 들기에 더 구현하기도 편한 합병정렬을 구현하기 위해 merge함수, merge\_sort함수를 별도로 구현하였으나 이 역시 모종의 이유로 실패하였습니다. 그 다음 정렬이 안되었지만 일단 DICT\_OUT파일에 넣기 위해 FILE포인터를 통해 다시 열어 fprintf함수는 printf처럼 파일 안에 쓰는게 가능한 함수이기에 이를 이용하여 정렬되었을 것이라고 예상된 dict배열을 집어넣고 fclose를 통해 종료하였습니다.

밑의 프로그램 수행 화면을 보시면 아시다 싶이 모종의 이유로 segmentation fault가 발생하였고, 저 오류가 발생하기 이전 다른 시도에서 정렬이 되긴 하였으나 파일을 잘못읽었는지 모두 0으로 나와 정렬이 실패하였다고 하여 이 해결방법을 여러가지 시도하던 도중 결국은 아래와 같은 결과가 나왔습니다.

1.2

s1910783@oak:hw09$ ./sort-test

Segmentation fault

끝.