/\*------------------------------------------------------------------------------------------------------------

C-Programming Asignment 4-2 : 지원인원을 선발 및 관리 해주는 프로그램.

학번 : B611019

전공 : 컴퓨터공학과 2학년

이름 : 김남일

날짜 : 2019년 06월 04일

주요 내용 :

4-2)정시에서 30명 선발하기.

지원자 1000명인데 그중에서 30명을 뽑아야한다. 이때 지원자 정보를 구조체배열을 선언해

지원번호, 이름, 수학, 영어, total 5개의 인수를 갖는다. 이때 각 정보들은

지원번호: 1에서 1000까지

이름: 성씨 - 대문자(A-Z), 이름 - 소문자(단, 이름은 최소3개 최대 5개의 길이를 가진다.)

수학 : 50 +- 30; 영어 : 60 +- 30; total : 수x0.6 + 영x0.4로 넣어준다.

정보 설정방법은 지원번호는 for(i=1;i<=1000;i++)에서 i값을 넣어주면 되지만, 그 외 나머지는 rand값을 넣어준다.

이름같은 경우는 for문 안에서 for (j = 1; j < (rand() % 3) + 4; j++)를 통해서 길이또한 rand로 설정한다.

예시) 지원번호: 2번 - 이름: Kfdgk , 수학점수: 76.000000, 영어점수: 56.000000, 전체점수: 68.000000

출력list는 4개가 있다.

#1 10명 지원자정보를 출력한다.

#2 통계자료 출력하기 (수학, 영어, total의 평균과 표준편차)

#3 상위 10명 성적((단, 만약 total점수가 같을시에 수학 > 영어 > 지원번호가 빠른순서로 순위를 정해준다.)

#4 합격자의 이름(단, 알파벳 순으로 정렬되고 다른 정보들은 보여주지 않는다.)

출력list가 끝나면 검색서비스를 지원여부를 묻는다. 검색서비스를 사용할 시에는

이름을 search문자열에 입력받고, b\_search함수를 통해서 이진탐색을 해준다. 이때 strcmp함수를 이용해서 문자열을 비교한다.

1) 이름이 존재시에는 모든정보전부다 출력해야한다. 동명이인들의 정보도 모두 줄력해준다.

(단, 이진탐색은 이름순으로 sorting한 자료를 가지고 하는것이 전제조건이다.)

2) 이름이 없다면 그 전, 후의 정보를 출력해주면서 그 사람은 없다고 알려주어야한다.

ex) 김남일을 검색시, 김남영/김남둘의 정보를 출력해주고 김남일씨는 없습니다.

사용한 함수들과 세부설명:

1. make\_list(struct student s[])함수를 통해서 1000명의 list값을 번호순으로 random하게 넣어준다.

2. copy\_list(struct student s1[], struct student s2[]) make\_list함수를 통해서 만든 원본정보들을 그대로 복사한다.

3. print\_list1(struct student s[]) 10명의 인적사항을 출력해준다.

4. print\_list2(struct student s[]) 전체 통계자료 출력해준다. 수학과 영어, total의 평균, 표준편차이다.

평균은 전체 합/전체 수이고, 분산은 (변수-평균)^2/전체 수이다. 표준편차는 sqrt(분산)으로 구현했다.

5. print\_list3(struct student s[]) 점수를 정렬한 후 출력해준다. 정렬을 할때 먼저

if (s[j].total > s[least].total) least = j; if (s[j].total == s[least].total && s[j].math > s[least].math) least = j; 처럼 최소값을 구하고 temp = s[i]; s[i] = s[least]; s[least] = temp;를 통해서 값을 바꿔준다(temp도 구조체이다.)

6. print\_list4(struct student s[]) 합격자 30명의 이름을 출력해준다.(점수순으로 sort하고 그 30명만 알파벳순으로 sort한다.)

7. name\_sort(struct student s[], int size) 앞의 첫 글짜(s[i].name[0])부터 마지막 글짜 까지 비교하면서 최솟값을 구해주고

그 값을 print\_list3함수 처럼 다시 치환해준다.

8. b\_search(struct student s[], int size, char search[]) 함수는 입력된 search문자열을 받아서 strcmp로 문자열을 비교한다. while (low <= high)동안 이름을 탐색하는데 만약 찾는 이름이 있다면 m값으로 return된다. 하지만 찾는 이름이 없다면 return (1000\*low) + high이 되는데 그 이유는 이진탐색을 활용하기 위해서 return값에 2개의 값을 담은 것이다.

이렇게 해주면 예를 들어 Knami"l"라는 사람을 검색했을때 만약 Knami"m"이라는 사람이 있다면 Knami까지는 접근하고 값이 return 되므로 high값이 바로 전 사람, low값이 바로 다음 사람을 나타낸다.

9. search\_name(struct student s[], int m)함수는 들어온 m값을 통해서 만약 m이 [1,1000]이라면 그 번호를 받아서

s[m].name에 접근하고 if (strcmp(s[m + a].name, s[m].name) == 0)을 통해서 m보다 위의 숫자에,

if (strcmp(s[m - a].name, s[m].name) == 0)를 통해서 m보다 낮은 숫자에 같은 이름을 가진 사람이 있는지 찾고. 있으면 for (i = min; i <= max; i++)를 통해서 동명이인 모두 출력해준다.

이때 단 m값이 [1,1000]의 숫자가 아니라면 b\_search에서 (1000\*low) + high값을 받아오는데 과제 4-1(b)의 (10\*s)+b 처럼 call\_by\_value를 통해서 2개의 값을 반환해준 것이다. 이렇게 해주면 low는 바로 다음사람, high는 바로 전 사람을의 값을 가지므로, s[m%1000]은 검색한 이름의 바로 이전사람, s[m/1000]은 검색한 이름의 바로 다음사람의 정보를 가진다.

------------------------------------------------------------------------------------------------------------\*/

#include <stdio.h>

#include <time.h>

#include <math.h>

#include <string.h>

struct student {

int num; //지원번호

char name[6]; //이름

double math; //수학

double eng; //영어

double total; //total

struct student \*p; //구조체 포인터

};

void make\_list(struct student s[]) { //1000명의 list를 만들어준다.

int len;

int i, j;

(unsigned)srand(time(NULL));

for (i = 1; i <= 1000; i++) {

s[i].num = i; //지원번호

s[i].name[0] = (rand() % 26) + 65; //성은 항상 대문자

for (j = 1; j < (rand() % 3) + 4; j++) {

s[i].name[j] = (rand() % 26) + 97;

}

s[i].name[j] = '\0';

s[i].math = (rand() % 61) + 20;// 20점에서 80점까지

s[i].eng = (rand() % 61) + 30; // 30점에서 90점까지

s[i].total = (s[i].math \* 0.6) + (s[i].eng \* 0.4); // 반영비율은 수학 60퍼, 영어 40퍼이다.

}

}

void copy\_list(struct student s1[], struct student s2[]) {

for (int i = 0; i <= 1000; i++)

s2[i] = s1[i];

}

void print\_list1(struct student s[]) { //10명의 인적사항출력해주기.

for (int i = 1; i <= 10; i++)

printf("지원번호:%2d번 - 이름: %6s , 수학점수: %f, 영어점수: %f, 전체점수: %f \n", s[i].num, s[i].name, s[i].math, s[i].eng, s[i].total);

printf("\n");

}

void print\_list2(struct student s[]) { //전체 통계자료 출력해주기.수학과 영어, total의 평균, 표준편차

int i;

double sum1 = 0, mean1 = 0;

double sum2 = 0, mean2 = 0;

double sum3 = 0, mean3 = 0;

double var1 = 0, var2 = 0, var3 = 0;

for (i = 1; i <= 1000; i++)

sum1 += s[i].math;

mean1 = (sum1 / 1000); //수학의 평균.

for (i = 1; i <= 1000; i++)

sum2 += s[i].eng;

mean2 = (sum2 / 1000); //영어의 평균.

for (i = 1; i <= 1000; i++)

sum3 += s[i].total;

mean3 = (sum3 / 1000); //전체의 평균.

for (i = 1; i <= 1000; i++) //수학의 표준편차

var1 += (s[i].math - mean1)\*(s[i].math - mean1);

var1 = (var1 / 1000);

for (i = 1; i <= 1000; i++) //영어의 표준편차

var2 += (s[i].eng - mean2)\*(s[i].eng - mean2);

var2 = (var2 / 1000);

for (i = 1; i <= 1000; i++) //전체의 표준편차

var3 += (s[i].total - mean3)\*(s[i].total - mean3);

var3 = (var3 / 1000);

printf("수학 평균: %f, 수학 표준편차 : %f \n", mean1, sqrt(var1));

printf("영어 평균: %f, 영어 표준편차 : %f \n", mean2, sqrt(var2));

printf("전체 평균: %f, 전체 표준편차 : %f \n", mean3, sqrt(var3));

printf("\n");

}

void print\_list3(struct student s[]) { //점수를 정렬한 후 출력해준다.

struct student temp;

int i, j, least;

for (i = 1; i <= 1000; i++)

{

least = i;

for (j = i + 1; j <= 1000; j++) {

if (s[j].total > s[least].total)

least = j;

if (s[j].total == s[least].total && s[j].math > s[least].math)

least = j;

if (s[j].total == s[least].total && s[j].math == s[least].math && s[j].eng > s[least].eng)

least = j;

if (s[j].total == s[least].total && s[j].math == s[least].math && s[j].eng == s[least].eng && s[j].num < s[least].num)

least = j;

}

temp = s[i];

s[i] = s[least];

s[least] = temp;

}

for (i = 1; i <= 10; i++)

printf("%2d등의 점수: %f \n", i, s[i].total);

printf("\n");

}

void print\_list4(struct student s[]) { //list3에 의해서 이미 점수순으로 출력되었음으로 앞의 30명의 이름만

int i;

printf("합격자 30명의 명단: \n");

for (i = 1; i <= 10; i++)

printf(" %6s ", s[i].name);

printf("\n");

for (i = 11; i <= 20; i++)

printf(" %6s ", s[i].name);

printf("\n");

for (i = 20; i <= 30; i++)

printf(" %6s ", s[i].name);

printf("\n");

}

void name\_sort(struct student s[], int size) { //이름을 알파벳순으로 정렬해준다.

struct student temp;

int i, least;

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

{

least = i;

for (int j = i + 1; j <= size; j++) { //성씨를 기준으로 알파벳순 정렬.

if (s[j].name[0] < s[least].name[0])

least = j;

else if (s[j].name[0] == s[least].name[0] && s[j].name[1] < s[least].name[1]) //성씨가 같으면 그다음 이름순으로

least = j;

else if (s[j].name[0] == s[least].name[0] && s[j].name[1] == s[least].name[1] && s[j].name[2] < s[least].name[2])

least = j;

else if (s[j].name[0] == s[least].name[0] && s[j].name[1] == s[least].name[1] && s[j].name[2] == s[least].name[2] && s[j].name[3] < s[least].name[3])

least = j;

else if (s[j].name[0] == s[least].name[0] && s[j].name[1] == s[least].name[1] && s[j].name[2] == s[least].name[2] && s[j].name[3] == s[least].name[3] && s[j].name[4] < s[least].name[4])

least = j;

else if (s[j].name[0] == s[least].name[0] && s[j].name[1] == s[least].name[1] && s[j].name[2] == s[least].name[2] && s[j].name[3] == s[least].name[3] && s[j].name[4] == s[least].name[4] && s[j].name[5] < s[least].name[5])

least = j;

}

temp = s[i];

s[i] = s[least];

s[least] = temp;

}

}

int b\_search(struct student s[], int size, char search[]) //찾는 이름을 이진탐색해준다 (단, 이미 sorting되어 있어야한다.)

{

int low, high, m;

low = 1;

high = size;

m = 0;

while (low <= high)

{

m = (low + high) / 2;

if (strcmp(s[m].name, search) == 0)

return m;

else if (strcmp(s[m].name, search) == 1)

high = m - 1;

else

low = m + 1;

}

return (1000\*low) + high; // 이진탐색을 활용하기 위해서 return값에 2개의 값을 담았다.

// 이렇게 해주면 예를들어 Knamil라는 사람을 검색했을때 만약 Knamim이라는 사람이 있다면

} // Knami까지는 접근하고 값이 return 되므로 high값이 바로 전 사람, low값이 바로 다음 사람을 나타낸다.

void search\_name(struct student s[], int m){ // 이름이 존재시 , 동일 인물까지 모두 출력해준다.

int i=0;

int max = m;int min = m;

int a = 1;

if (m <= 1000) // 이름을 찾고 그 위, 아래로 흝어면서 동일인물이 있나없나 체크한다.

{

while (1)

{

if (strcmp(s[m + a].name, s[m].name) == 0)

++a;

else if (strcmp(s[m + a].name, s[m].name) != 0)

break;

}

max = m + a - 1;

min = m;

a = 1;

while (1)

{

if (strcmp(s[m - a].name, s[m].name) == 0)

a++;

else if (strcmp(s[m - a].name, s[m].name) != 0)

break;

}

min = m - a + 1;

for (i = min; i <= max; i++)

printf("지원번호:%2d번 - 이름: %6s , 수학점수: %f, 영어점수: %f, 전체점수: %f \n", s[i].num, s[i].name, s[i].math, s[i].eng, s[i].total);

}

else { //이름을 검색했는데 없어도 그 근처까지 low와 high가 이동하므로 그 점을 이용한다 (과제 4-1(b)의 스트라이크\_볼게임 처럼 call\_by\_value로 2개이상 값 반환하기.)

printf("찾으시는 이름이 없습니다.\n");

printf("지원번호:%2d번 - 이름: %6s , 수학점수: %f, 영어점수: %f, 전체점수: %f \n", s[m % 1000].num, s[m % 1000].name, s[m % 1000].math, s[m % 1000].eng, s[m % 1000].total); //바로 전사람

printf("지원번호:%2d번 - 이름: %6s , 수학점수: %f, 영어점수: %f, 전체점수: %f \n", s[m / 1000].num, s[m / 1000].name, s[m / 1000].math, s[m / 1000].eng, s[m / 1000].total); //바로 다음사람

}

}

int main() {

struct student s1[1001]; //원본파일.

struct student s2[1001]; //복사파일.

char answer = {0};

char search[7] = "";

int m;

struct student \*p;

make\_list(s1); //1000명의 지원자를 받는다.

copy\_list(s1, s2);//파일을 복사한다.

print\_list1(s2); //10명의 지원정보를 출력해준다.

print\_list2(s2); //점수들의 평균과 표준편차를 출력해준다.

print\_list3(s2); //점수순으로 10명을 출력해준다.

name\_sort(s2, 30);//앞의 상위 30명의 사람들을 이름순으로 다시 정렬해준다.

print\_list4(s2); //합격자를 이름순으로 30명 출력해준다. 이름 65에서 90은 대문자, 97에서 122는 소문자이다.

name\_sort(s2, 1000);

while (1)

{

printf("이름을 검색하시겠습니까?(y or n) :");

scanf(" %c", &answer);

if ( answer == 'y' )

{

printf("이름을 입력하시오: ");

scanf(" %s", search);

m = b\_search(s2, 1000, search);

printf("m은 %d이다. \n", m);

search\_name(s2, m);

printf("\n");

}

else break;

} return 0;

}