/\*

240104 알고리즘 1번 문제

데이터베이스 argv활용하기

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

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

{

int a, b, result, answer;

int correct = 0, i = 0;

FILE \*infile;

if (argc != 2) {

printf("실행인수를 잘못 주었습니다...\n");

exit(1);

}

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

printf("입력 파일을 열 수 없습니다. \n");

exit(1);

}

while (fscanf(infile, "%d %d", &a,&b) != EOF) {

printf("%d : gcd(&d, &d) = ", i + 1, a, b);

scanf("%d", &answer);

result = gcd(a, b);

if (answer != result)

{

printf("Answer : gcd(%d, %d) = %d \n", a, b, result);

}

else {

printf("Correct...\n");

correct++;

}

i++;

}

printf("%d 문제 중에 %d 문제 통과하셨습니다...\n,i,correct");

}

int gcd(int x, int y) // 재귀적 정의법

{

if (y == 0)

return x;

if (x == 0)

return y;

else

return gcd(y, (x % y));

}

/\*

240104 알고리즘 2번 문제

하노이 탑

입력 : 디스크의 개수, 출발 막대, 목적 막대, 중간 막대

출력 : 각 단계에서 디스크가 어느 막대로부터 어느 막대로 움직이는지 출력

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

void hanoi(int i, char a, char c, char b)

{

if (i == 1)

printf("Move disk from %c to %c\n", a, c);

else {

hanoi(i - 1, a, b, c);

hanoi(1, a, c, b);

hanoi(i - 1, b, c, a);

}

}

void main()

{

int n;

printf("하노이의 탑 프로그램입니다 \n");

printf("움직일 원반의 개수를 입력해주세요\n");

scanf("%d", &n);

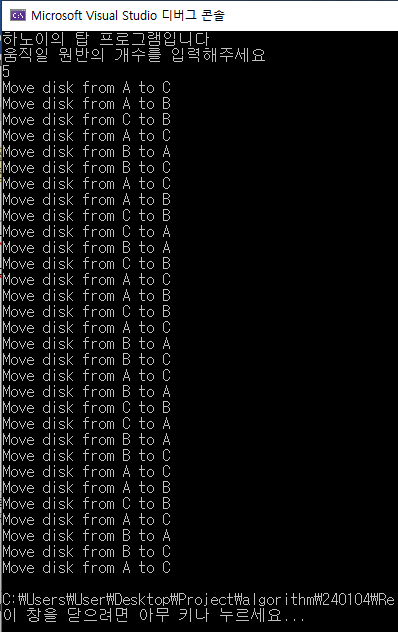
if (n <= 0)

printf("\n디스크의 개수를 다시 설정해주세요\n");

else

hanoi(n, 'A', 'C', 'B');

}



/\*

240104 알고리즘 3번 문제

TOP-Down approach 설계 - stepwise Refinement

1. 문제정의 단계에서 얻은 문제해결방안을 바탕으로 Tree 형태로 문제를 분할한다

2. 하위레벨로 내려 올 수 록 구체화 된다.

3. 설계된 작업을 알고리즘으로 작성한다.

4. 최종적으로 필요한 함수의 종류를 마련하고 프로그램코드로 표현될 때까지 세분화한다.

[선택정렬]

가. n개의 데이터로 구성되어있는 정수 데이터 집합의 데이터를 오름차순으로 정렬한다.(작은 값 -> 큰 값)

정렬되지 않은 정수들 중에서 가장 작은 값을 찾아서 이미 정렬도니 리스트의 다음자리에 배치

입력 : 데이터의 수(n), 정수 배열(list)

출력 : 정렬된 정수배열(list)

단계 1. 배열 list에 입력 데이터 n개 저장 (list[0], list[1]...)

단계 2. s= 0, 정렬된 데이터가 처음에는 없다고 시작

단계 3. s = n-1 이면 모든 데이터가 정렬되어 있으므로 단계 6으로 간다. 아니면 s부터 n-1의 위치에 있는

데이터중 최소값을 찾는다(m의 위치라 가정한다)

단계 4. list[m]은 list[s]라 가정한다

단계 5.

단계 6.

배열 list에 입력데이터를 받고 selection\_sort(int list[],int n)에 의해 정렬하고

그 결과를 출력하는 함수 print\_data(int list[], int n)을

main()에서 사용하도록 프로그램을 작성하여 오름차순으로 또한 내림차순으로 실행시켜보시오

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

void as\_selection\_sort(int list[], int n)

{

int s, m, j, temp;

for (s = 0; s < n - 1; s++) {

m = s;

for (j = s + 1; j < n; j++)

if (list[j] < list[m]) m = j;

temp = list[s];

list[s] = list[m];

list[m] = temp;

}

}

void de\_selection\_sort(int list[], int n)

{

int s, m, j, temp;

for (s = 0; s < n - 1; s++) {

m = s;

for (j = s + 1; j < n; j++)

if (list[j] > list[m]) m = j;

temp = list[s];

list[s] = list[m];

list[m] = temp;

}

}

void print\_data(int a[], int n){

int i;

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

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

}

}

void main()

{

int num;

int list[100];

int cond;

printf("선택정렬 프로그램입니다. \n");

printf("정렬할 숫자의 개수를 입력해주세요. \n");

scanf("%d", &num);

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

printf("숫자를 입력해주세요\n");

scanf("%d", &list[i]);

}

printf("오름차순은 1, 내림차순은 2를 입력해주세요 \n");

scanf(" %d", &cond);

printf("정렬을 시작합니다... \n");

if (cond = 1)

as\_selection\_sort(list, num);

print\_data(list, num);

exit(1);

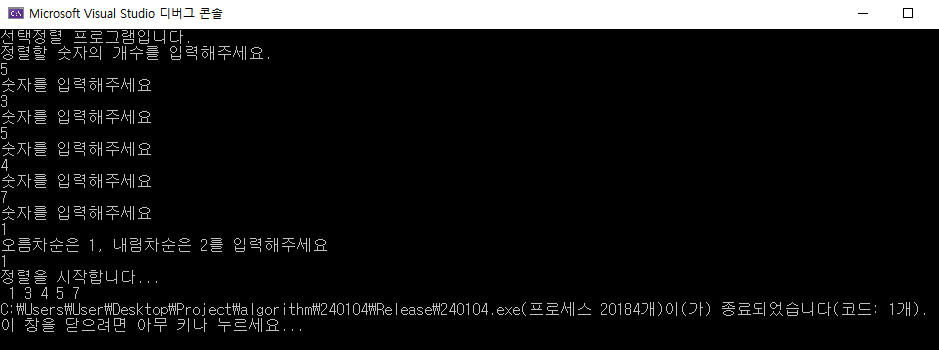
if (cond = 2)

de\_selection\_sort(list, num);

print\_data(list, num);

exit(1);

}



/\*

240104 알고리즘 4번 문제 - 이진검색

TOP-Down approach 설계 - stepwise Refinement

오름차순으로 정렬된 n개의 데이터로부터 주어진 어떤 값이 있는지 검색한다.

중앙값과 비교하여 검색 구간을 줄여간다.

입력: 정렬된 n개의 정수 배열(list), list[0], list[1],..list[n-1], 찾으려는 값(key)

출력: 검색결과

1. 입력데이터를 준비

2. 검색구간을 설정한다

3. 검색구간에 데이터가 있으면 다음을 수행하고 없으면 단계 4 ->중앙값과 찾으려는 key값이 같으면 그 위치 출력

- 중앙값과 찾으려는 key값이 같으면 그 위치를 출력한다.

- 같지 않으면 단계 2로 간다

4. 주어진 데이터 집합 안에 key가 존재하지 않으므로 -1 출력

중앙값 인덱스는 n/2 + 1

입력데이터 준비 - n,list,key

검색구간결정 : left, right

중앙값과 비교 key: list[mid]

1. 내림차순으로 정렬한 데이터를 가지고 있는 경우, 이진검색 알고리즘의 어떤 부분을 바꿔야 하는가?

답) key < list[mid] 인 경우 : left = mid +1 / key > list[mid] 인 경우 : right = mid - 1

2. 이진 검색 알고리즘에 따라 시뮬레이션을 하라 -> 표를 이용

3. main()함수를 작성하여 실행 결과를 얻어보자

배열 list에 입력데이터를 받고 찾고자 하는 key 값을 입력받는다.

또한 정렬되었는지 물어보며

정렬되지 않은 데이터라면 선택정렬을 이용해 정렬하고 bsearch를 이용해 key값이 몇번째 데이터인지 출력,

아니라면 not exist 출력으로 종료

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

int bsearch\_as(int a[], int n, int key) {

int mid;

int left = 0, right = n - 1;

while (left <= right) {

mid = (left + right) / 2;

switch ( compare(key, a[mid])) {

case '>': left = mid + 1; break;

case '<': right = mid + 1; break;

case '=': return mid;

} //switch

} // while

return -1;

}

/\*

if (key > a[mid]) left = mid + 1;

else if (key < a[mid]) right = mid - 1;

else return mid;

\*/

void as\_selection\_sort(int list[], int n)

{

int s, m, j, temp;

for (s = 0; s < n - 1; s++) {

m = s;

for (j = s + 1; j < n; j++)

if (list[j] < list[m]) m = j;

temp = list[s];

list[s] = list[m];

list[m] = temp;

}

}

void de\_selection\_sort(int list[], int n)

{

int s, m, j, temp;

for (s = 0; s < n - 1; s++) {

m = s;

for (j = s + 1; j < n; j++)

if (list[j] > list[m]) m = j;

temp = list[s];

list[s] = list[m];

list[m] = temp;

}

}

void print\_data(int a[], int n) {

int i;

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

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

}

}

void main()

{

int num;

int list[100];

int key;

int n;

int kind;

int result1;

int cond;

char res;

printf("이진탐색 프로그램입니다. \n");

printf("정리할 숫자의 개수를 입력해주세요. \n");

scanf("%d", &num);

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

printf("숫자를 입력해주세요\n");

scanf("%d", &list[i]);

}

printf("이 숫자들이 정렬되었습니까?(Y/N)\n");

scanf(" %c", &res);

if (res == 'N') {

printf("선택정렬을 진행합니다..\n");

printf("오름차순은 1, 내림차순은 2를 입력하세요 \n");

scanf(" %d", &cond);

if (cond == 1){

as\_selection\_sort(list, num);

printf("오름차순 완료 \n");

print\_data(list, num);

}

if (cond == 2)

de\_selection\_sort(list, num);

printf("내림차순 완료 \n");

print\_data(list, num);

}

printf("\n 이진탐색을 진행합니다.. \n");

printf("찾고자 하는 값을 입력하세요.. \n");

scanf("%d", &key);

}

/\*

240104 알고리즘 5번 문제 - 계산기 338p

상태도 분석 기법 -> 상태 전이도를 통하여 분석 확장

정수 단순 계산기 프로그램

stage 1 -> stage 2 -> stage 3

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#define STAGE1 0

#define STAGE2 1

#define STAGE3 2

void real\_calculator(void);

double real\_operation(char op, double operand1, double operand2);

double get\_real(double current\_operand, char c, int status); // 이해해야하는 중요 함수

int is\_operator(char c);

int is\_digit(char c);

void main()

{

printf("계산기 프로그램을 시작합니다. \n");

printf("실수형의 피연산자와 사칙연산자 중 하나를 입력하십시오. \n");

real\_calculator();

printf("\n계산기 프로그램을 종료합니다. \n");

}

int is\_operator(char c) {

if (c == '+' || c == '-' || c == '\*' || c == '/')

return 1;

else

return 0;

}

int is\_digit(char c) {

if (c >= '0' && c <= '9')

return 1;

else

return 0;

}

void real\_calculator(void)

{

double operand1 = 0, operand2 = 0, result;

char op, c;

int current\_stage = STAGE1, status = 0;

while ((c = getchar()) != 'x') {

if (is\_digit(c)) {

operand1 = get\_real(operand1, c, status);

if (status != 0) status++;

}

else if (is\_operator(c)) {

op = c;

current\_stage = STAGE2;

status = 0;

operand2 = 0;

}

else if (current\_stage == STAGE2) {

if (is\_digit(c)) {

operand2 = get\_real(operand2, c, status);

if (status != 0) status++;

}

else if (c == '.') status = 1;

else if (c == '\n') { // 중요부분

result = real\_operation(op, operand1, operand2);

printf("= %.3f", result);

current\_stage = STAGE3;

status = 0;

}

} //STAGE2

else if (current\_stage == STAGE3) {

if (c == '\n') {

operand1 = 0;

current\_stage = STAGE1;

}

else if (is\_operator(c)) {

op = c;

current\_stage = STAGE2;

}

}

}

}

double real\_operation(char op, double operand1, double operand2)

{

double result;

switch (op) {

case '+' :

result = operand1 + operand2;

break;

case '-':

result = operand1 - operand2;

break;

case '\*':

result = operand1 \* operand2;

break;

case '/':

if (operand2 !=0) result = operand1 / operand2;

break;

}

return result;

}

결과 : 외부 기호 참조 오류 발생

