**Lab 05**

|  |
| --- |
| 그림입니다. 원본 그림의 이름: YU_UI_RGB-10.png 원본 그림의 크기: 가로 2256pixel, 세로 3047pixel 프로그램 이름 : Adobe ImageReady |

|  |  |
| --- | --- |
| 과목명 | 프로그래밍 언어 |
| 교수님 | 김영탁 교수님 |
| 이 름 | 김주환 |
| 학 번 | 21812158 |
| 일 자 | 2021.04.01.목 |

**5.1 다중 소스파일 : BigArray.h, BigArray.c**

// BigArray.h

#ifndef BIG\_ARRAY\_H // 헤더 중복 방지

#define BIG\_ARRAY\_H

#include <stdio.h>

#include <math.h>

#include <time.h>

void printBigArraySample(int\* array, int size, int line\_size, int num\_sample\_lines);

void fprintBigArraySample(FILE\* fout, int\* array, int size, int line\_size);

void genBigRandArray(int\* array, int size);

void suffleArray(int\* array, int size);

void genArrayStatistics(int\* array, int size);

void fgetArrayStatistics(FILE\* fout, int\* array, int size);

#endif

1) BigArray.h는 다음과 같이 기호상수와 배열 관련 함수들의 함수원형등을 포함한다.

2) BigArray.c에 배열의 출력과 통계처리를 위한 다음 함수들을 구현하라.

// 실행창에 배열 정보 출력

void printBigArraySample(int\* array, int size, int line\_size, int num\_sample\_lines) {

int count = 0; // 몇번째 수인지 확인

int brk\_temp = 0; // 센티널을 만나면 탈출

for (int i = 0; i < num\_sample\_lines; i++) { // 기본 배열 출력

for (int j = 0; j < line\_size; j++) { // line\_size개 만큼 num\_sample\_lines줄 출력

if (count > size) { // 지정한 크기 초과하면 탈출

printf("\n");

return;

}

if (array[count] == -1) { // 센티널 만나면 탈출

brk\_temp++;

break;

}

printf("%8d", array[count]);

count++;

}

printf("\n");

if (brk\_temp == 1)break;

}

if (20 < size && size < 41) { // 20 초과 40 이하인 경우의 배열 출력

count = num\_sample\_lines \* line\_size; // 첫번째 출력 이후로 초기값 재설정

brk\_temp = 0;

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

for (int j = 0; j < line\_size; j++) {

if (count > size) {

printf("\n");

return;

}

if (array[count] == -1) {

brk\_temp++;

break;

}

printf("%8d", array[count]);

count++;

}

printf("\n");

if (brk\_temp == 1)break;

}

}

else if(40<size) { // 40 초과인 경우의 배열 출력

brk\_temp = 0;

if (count < (size - line\_size \* num\_sample\_lines)) // 마지막 num\_sample\_lines줄 만큼 출력

count = size - (line\_size \* num\_sample\_lines); // 하도록 초기값 재설정

if (count >= size) return;

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

brk\_temp = 0;

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

for (int j = 0; j < line\_size; j++) {

if (count > size) {

printf("\n");

return;

}

if (array[count] == -1) {

brk\_temp++;

break;

}

printf("%8d", array[count]);

count++;

}

printf("\n");

if (brk\_temp == 1)break;

}

printf("\n");

}

}

// 지정된 파일에 배열 정보 출력

void fprintBigArraySample(FILE\* fout, int\* array, int size, int line\_size, int num\_sample\_lines) {

int count = 0; // 몇번째 수인지 확인

int brk\_temp = 0; // 센티널 만나면 탈출

for (int i = 0; i < num\_sample\_lines; i++) { // 기본 배열 출력

for (int j = 0; j < line\_size; j++) {

if (count > size) { // 지정된 크기를 초과하면 탈출

fprintf(fout, "\n");

return;

}

if (array[count] == -1) { // 센티널 만나면 탈출

brk\_temp++;

break;

}

fprintf(fout, "%8d", array[count]);

count++;

}

fprintf(fout, "\n");

if (brk\_temp == 1)break;

}

if (20 < size && size < 41) { // 20 초과 40 이하인 경우의 배열 출력

count = num\_sample\_lines \* line\_size; // 첫 배열 출력 이후의 값으로 초기값 재설정

brk\_temp = 0;

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

for (int j = 0; j < line\_size; j++) {

if (count > size) {

fprintf(fout, "\n");

return;

}

if (array[count] == -1) {

brk\_temp++;

break;

}

fprintf(fout, "%8d", array[count]);

count++;

}

fprintf(fout, "\n");

if (brk\_temp == 1)break;

}

}

else if (40 < size) { // 40 초과인 경우의 배열 출력

brk\_temp = 0;

if (count < (size - line\_size \* num\_sample\_lines)) // 마지막 부분 출력하도록

count = size - (line\_size \* num\_sample\_lines); // 초기값 재설정

if (count >= size) return;

fprintf(fout, "\n . . . . .\n\n");

brk\_temp = 0;

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

for (int j = 0; j < line\_size; j++) {

if (count > size) {

fprintf(fout, "\n");

return;

}

if (array[count] == -1) {

brk\_temp++;

break;

}

fprintf(fout, "%8d", array[count]);

count++;

}

fprintf(fout, "\n");

if (brk\_temp == 1)break;

}

fprintf(fout, "\n");

}

}

// 배열의 속성을 지정된 파일에 출력

void fgetArrayStatistics(FILE\* fout, int\* array, int size) {

int data, min, max;

double sum = 0.0, var, diff, sq\_diff\_sum = 0.0, avg, std\_dev;

min = INT\_MAX;

max = INT\_MIN;

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

data = array[i];

sum += data;

if (data < min)min = data; // 최소값

if (data > max)max = data; // 최대값

}

avg = sum / (double)size; // 평균

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

diff = array[i] - avg;

sq\_diff\_sum += diff \* diff;

}

var = sq\_diff\_sum / (double)size; // 분산

std\_dev = sqrt(var); // 표준편차

fprintf(fout, "min (%3d), max (%3d), ", min, max);

fprintf(fout, "sum (%8.2lf), average (%8.2lf), ", sum, avg);

fprintf(fout, "variance (%8.2lf), standard deviation (%8.2lf)\n", var, std\_dev);

}

// 배열의 속성을 실행창에 출력

void getArrayStatistics(int\* array, int size) {

int data, min, max;

double sum = 0.0, var, diff, sq\_diff\_sum = 0.0, avg, std\_dev;

min = INT\_MAX;

max = INT\_MIN;

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

data = array[i];

sum += data;

if (data < min)min = data; // 최소값

if (data > max)max = data; // 최대값

}

avg = sum / (double)size; // 평균

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

diff = array[i] - avg;

sq\_diff\_sum += diff \* diff;

}

var = sq\_diff\_sum / (double)size; // 분산

std\_dev = sqrt(var); // 표준편차

printf("min (%3d), max (%3d), ", min, max);

printf("sum (%8.2lf), average (%8.2lf), ", sum, avg);

printf("variance (%8.2lf), standard deviation (%8.2lf)\n", var, std\_dev);

}

// 30비트 난수 생성

void genBigRandArray(int\* array, int size) {

char\* flag; // 난수가 생성되었는지 확인

int count = 0;

unsigned int u\_int32 = 0;

unsigned int bigRand; // 생성된 난수값

flag = (char\*)calloc(size, sizeof(char)); // 동적 할당

while (count < size) {

u\_int32 = ((long)rand() << 15) | rand();

bigRand = u\_int32 % size;

if (flag[bigRand] == 1) {

continue;

}

else { // 이미 생성된 난수인지 확인

flag[bigRand] = 1;

array[count++] = bigRand;

}

}

free(flag); // 할당 해제

}

3) Data\_array.c

// Data\_array.c

#include "BigArray.h"

extern int data\_array[100] = {

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,

21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, -1

};

**5.2 파일 입출력**

1 2 3 4 5 6 7 8 9 10

11 12 13 14 15 16 17 18 19 20

-1

1) 데이터 파일 입력

while (fscanf(fin, "%d", &data) != EOF) { // 파일의 배열을 읽어온다

if (data == -1) break;

data\_array[num\_data] = data;

num\_data++;

}

2) 파일 출력

// 배열의 속성을 지정된 파일에 출력

void fgetArrayStatistics(FILE\* fout, int\* array, int size) {

int data, min, max;

double sum = 0.0, var, diff, sq\_diff\_sum = 0.0, avg, std\_dev;

min = INT\_MAX;

max = INT\_MIN;

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

data = array[i];

sum += data;

if (data < min)min = data; // 최소값

if (data > max)max = data; // 최대값

}

avg = sum / (double)size; // 평균

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

diff = array[i] - avg;

sq\_diff\_sum += diff \* diff;

}

var = sq\_diff\_sum / (double)size; // 분산

std\_dev = sqrt(var); // 표준편차

fprintf(fout, "min (%3d), max (%3d), ", min, max);

fprintf(fout, "sum (%8.2lf), average (%8.2lf), ", sum, avg);

fprintf(fout, "variance (%8.2lf), standard deviation (%8.2lf)\n", var, std\_dev);

}

// 30비트 난수 생성

void genBigRandArray(int\* array, int size) {

char\* flag; // 난수가 생성되었는지 확인

int count = 0;

unsigned int u\_int32 = 0;

unsigned int bigRand; // 생성된 난수값

flag = (char\*)calloc(size, sizeof(char)); // 동적 할당

while (count < size) {

u\_int32 = ((long)rand() << 15) | rand();

bigRand = u\_int32 % size;

if (flag[bigRand] == 1) {

continue;

}

else { // 이미 생성된 난수인지 확인

flag[bigRand] = 1;

array[count++] = bigRand;

}

}

free(flag); // 할당 해제

}

**5.3 동적 배열 생성 및 중복되지 않는 난수 생성**

**5.4 main.c 파일**

// main,c

#include <stdio.h>

#include <conio.h>

#include <math.h>

#include <time.h>

#include "BigArray.h"

#define ESC 0x1B

#define MAX\_NUM\_DATA 100

// #define TEST\_WITH\_INPUT\_DATA

void arrayStatistics\_basicArray(FILE\* fout);

void arrayStatistics\_externArray(FILE\* fout);

void arrayStatistics\_fileDataArray(FILE\* fout);

void arrayStatistics\_inputArray(FILE\* fout);

void arrayStatistics\_genBigRandArray(FILE\* fout);

#define Data\_Input\_File "Data\_input.txt" // 기존에 작성된 파일명

#define Data\_Output\_File "Data\_output.txt" // 결과를 표시할 파일명

int main(int argc, char argv[]) {

FILE\* fout;

char menu;

fout = fopen(Data\_Output\_File, "w"); // 파일 쓰기모드로 열기

if (fout == NULL) { // 예외처리

printf("Error in creation of %s !!\n", Data\_Output\_File);

return -1;

}

while (1) {

printf("\nTest Array Handling (1 : data\_array; 2 : extern\_array; \

3 : data\_file; 4 : data\_input; 5 : genBigRandArray; Esc : terminate) : ");

menu = \_getche();

if (menu == ESC) break;

switch (menu) {

case '1':

arrayStatistics\_basicArray(fout); // 기본 배열에 대한 정보

break;

case '2':

arrayStatistics\_externArray(fout); // 외부 배열에 대한 정보

break;

case '3':

arrayStatistics\_fileDataArray(fout); // 이미 생성된 파일 내부의 배열에 대한 정보

break;

case '4':

arrayStatistics\_inputArray(fout); // 실시간으로 입력한 배열에 대한 정보

break;

case '5':

arrayStatistics\_genBigRandArray(fout); // 무작위로 생성한 배열에 대한 정보

break;

default:

break;

}

}

fclose(fout); // 파일 닫기

return 0;

}

void arrayStatistics\_basicArray(FILE\* fout) { // 기본 배열

int num\_data = 10;

int data\_array[MAX\_NUM\_DATA] = { 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, -1 };

printf("\nArrayStatistics\_basicArray .....\n");

fprintf(fout, "\nArrayStatistics\_basicArray .....\n");

printBigArraySample(data\_array, num\_data, 10, 2);

fprintBigArraySample(fout, data\_array, num\_data, 10, 2);

getArrayStatistics(data\_array, num\_data);

fgetArrayStatistics(fout, data\_array, num\_data);

printf("\narrayStatistics\_basicArray - completed. Result is also stored in output file(%s).\n", Data\_Output\_File);

}

void arrayStatistics\_externArray(FILE\* fout) { // 외부 배열

int num\_data = 0;

extern int data\_array[MAX\_NUM\_DATA];

printf("\nArrayStatistics\_externArray .....\n");

fprintf(fout, "\nArrayStatistics\_externArray .....\n");

for (int i = 0; i < MAX\_NUM\_DATA; i++) {

if (data\_array[i] == -1) break;

else num\_data++;

}

printBigArraySample(data\_array, num\_data, 10, 2);

fprintBigArraySample(fout, data\_array, num\_data, 10, 2);

getArrayStatistics(data\_array, num\_data);

fgetArrayStatistics(fout, data\_array, num\_data);

printf("\narrayStatistics\_externArray - completed. Result is also stored in output file(%s).\n", Data\_Output\_File);

}

void arrayStatistics\_fileDataArray(FILE\* fout) { // 지정된 파일의 배열

FILE\* fin;

int data, num\_data = 0;

int data\_array[MAX\_NUM\_DATA] = { 0 };

printf("\nArrayStatistics\_fileDataArray .....\n");

fprintf(fout, "\nArrayStatistics\_fileDataArray .....\n");

fin = fopen(Data\_Input\_File, "r"); // 읽기 모드로 파일 열기

for (int i = 0; i < MAX\_NUM\_DATA; i++) {

if (data\_array[i] == -1) break;

fprintf(fin, "%d", data\_array[i]);

}

while (fscanf(fin, "%d", &data) != EOF) { // 파일의 배열을 읽어온다

if (data == -1) break;

data\_array[num\_data] = data;

num\_data++;

}

fclose(fin);

printBigArraySample(data\_array, num\_data, 10, 2);

fprintBigArraySample(fout, data\_array, num\_data, 10, 2);

getArrayStatistics(data\_array, num\_data);

fgetArrayStatistics(fout, data\_array, num\_data);

printf("\narrayStatistics\_fileDataArray - completed. Result is also stored in output file(%s).\n", Data\_Output\_File);

}

void arrayStatistics\_inputArray(FILE\* fout) {

int num\_data, data;

int data\_array[MAX\_NUM\_DATA] = { 0 };

printf("\nArrayStatistics\_inputArray .....\n");

fprintf(fout, "\nArrayStatistics\_inputArray .....\n");

printf("Please input the number of integers (less than %d) = ", MAX\_NUM\_DATA);

scanf("%d", &num\_data);

printf("Input %d integer data : ", num\_data);

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

scanf("%d", &data);

data\_array[i] = data;

data\_array[i + 1] = -1;

}

printBigArraySample(data\_array, num\_data, 10, 2);

fprintBigArraySample(fout, data\_array, num\_data, 10, 2);

getArrayStatistics(data\_array, num\_data);

fgetArrayStatistics(fout, data\_array, num\_data);

printf("\narrayStatistics\_inputArray - completed. Result is also stored in output file(%s).\n", Data\_Output\_File);

}

void arrayStatistics\_genBigRandArray(FILE\* fout) {

int num\_data = 0;

int\* dyn\_array = NULL;

printf("\nArrayStatistics\_genBigRandArray .....\n");

fprintf(fout, "\nArrayStatistics\_genBigRandArray .....\n");

printf("Big Array Size (more than 50000) = ");

scanf("%d", &num\_data);

printf("Creating big random integer array (size : %d)\n", num\_data);

dyn\_array = (int\*)calloc(num\_data, sizeof(int));

if (dyn\_array == NULL) {

printf("Error in dynamic creation of big\_array (size = %d) !!!", num\_data);

exit(1);

}

genBigRandArray(dyn\_array, num\_data);

// 동적 배열 생성 및 중복되지 않는 난수 배열 생성 기능을 이곳에 구현할 것

printBigArraySample(dyn\_array, num\_data, 10, 2);

fprintBigArraySample(fout, dyn\_array, num\_data, 10, 2);

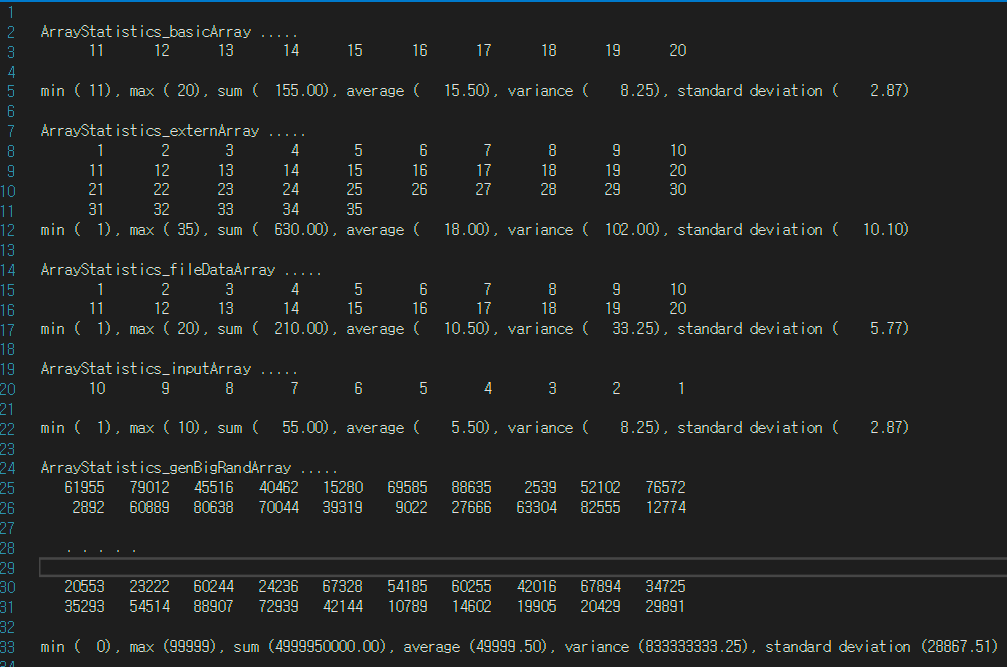
getArrayStatistics(dyn\_array, num\_data);

fgetArrayStatistics(fout, dyn\_array, num\_data);

// 동적 배열의 반환 기능을 이곳에 구현할 것

printf("\narrayStatistics\_genBigRandArray - completed. Result is also stored in output file(%s).\n", Data\_Output\_File);

}

**5.5 실행 결과**텍스트이(가) 표시된 사진

자동 생성된 설명

Oral Test

|  |
| --- |
| **(1) 다중 소스 파일 구조의 프로그램 개발에서 사용자 정의 헤더 파일에 포함되는 내용에 대하여 설명하라.**  사용자 정의 헤더 파일에는 기호상수, 함수원형, 구조체 등이 포함될 수 있다. 이는 반복되거나 필요이상으로 긴 내용이거나, main.c에 반드시 포함하지 않고도 충분히 이해할 수 있는 정보다. |

|  |
| --- |
| **(2) 전처리기 지시어를 사용하여 헤더 파일의 중복 포함을 방지하는 방법과 조건부 컴파일을 실행하는 방법을 각각 예를 들어 설명하라**   * 헤더 파일의 중복 포함을 방지하는 방법 헤더 파일을 처음 포함했을 경우, 헤더 파일 내부에서 특정 기호상수를 정의한다. 이후 중복하여 헤더 파일을 포함했을 경우, #ifndef를 통해 이전에 특정 기호상수를 정의하지 않았을 때만 헤더파일을 포함한다. 조건부 컴파일의 마지막에는 #endif로 끝을 맺어준다. * 조건부 컴파일을 실행하는 방법 헤더 파일을 이미 포함했다면, 중복을 방지하기위해 특정 기호상수를 정의했을 것이다. 그렇다면, #ifdef를 통해 특정 기호상수가 포함되었는지 확인해서 “이미 헤더파일을 포함했습니다” 라는 문구를 출력해준다. 이는 #endif로 끝난다. |

|  |
| --- |
| **(3) RAND\_MAX보다 큰 정수인 BIG\_RAND\_MAX까지의 난수를 중복되지 않게 생성하여 주어진 배열에 차례로 담아주는 genBigArray(int\* array, int size)의 동작 절차에 대하여 상세하게 설명하라.**   * 변수 생성 flag : 난수가 생성되었는지 확인하는 역할 count : 생성한 배열의 크기만큼 반복할 수 있도록 카운트하는 역할 u\_int32 : 30비트 크기의 난수를 저장하는 역할 bigRand : 30비트 난수의 크기를 원하는 크기로 재구성하는 역할 * 동적 할당 calloc을 통해 flag를 array 와 같은 크기로 동적할당 해준다. * 난수 생성 15bit의 난수를 생성한 후 15bit 왼쪽으로 이동해서 빈 자리에 다시 15bit 난수를 생성하고 두 난수를 bitOR 해주면 30bit 크기의 난수를 생성할 수 있다. 이후, % size를 통해 원하는 크기로 재구성해준다. * 이미 생성된 난수인지 확인 flag[BigRand]에 1이 들어 있다면 난수 생성과정을 다시 거쳐서 다른 난수를 생성하고, 0이 들어 있다면 1을 넣고 생성된 난수를 array[count]에 넣고 count++를 실행해준다. * 할당 해제 free(flag)를 통해 동적할당을 해제해준다.   추가로 offset(초기값)을 설정할 수 있는데, 이는 배열을 0부터 시작하는 것이 아니라 원하는 값부터 시작할 수 있도록 설정해준다. |

|  |
| --- |
| **(4) 주어진 배열의 통계적 특성을 산출하기 위하여 평균값, 분산, 표준편차를 계산하는 방법에 대하여 상세하게 설명하라.**   * 평균값(mean) : sum / size 1) 각 배열 원소의 값을 반복문을 통해 모두 더한다. 2) 위에서 더한 값을 배열의 크기로 나눈 값이 평균값이다. * 분산(variance) : sum of ((value – avg)^2) / size 1) 각 배열의 원소와 위에서 구한 평균값의 차를 구한다. 2) 해당 차를 각각 제곱해주고 모두 더한다. 3) 위의 합을 배열의 크기로 나눈 값이 분산이다. * 표준편차(standard deviation) : var^(1 / 2) 위의 분산에 루트를 취해준 것이 표준편차다. |