2021-1 프로그래밍언어 실습 5

5.1 다중 소스파일: BigArray.h, BigArray.c

1) BigArray.h는 다음과 같이 기호상수 (symbolic constant)와 배열 관련 함수들의 함수원형 (function prototype)들을 포함한다.

```
/* BigArray.h*/

#ifndef BIG_ARRAY_H
#define BIG_ARRAY_H

#include <stdio.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,
      int num_sample_lines);
void genBigRandArray(int *array, int size);
void suffleArray(int *array, int size);
void getArrayStatistics(int *array, int size);
void fgetArrayStatistics(FILE *fout, int *array, int size);
#endif
```

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

- void printBigArraySample(int *array, int size, int line_size,
  int num_sample_lines);
  // 주어진 배열을 한 줄에 line_size씩, 배열의 첫부분 num_sample_lines 줄, 마지막 부분의 num_sample_lines 줄을 출력
- void fprintBigArraySample(FILE *fout, int *array, int size, int line_size,
  int num_sample_lines);
  // 주어진 배열을 지정된 파일에 출력하며, 한 줄에 line_size씩, 배열의 첫부분 num_sample_lines 줄, 마지막 부분의 num_sample_lines 줄을 출력
- void getArrayStatistics(int *array, int size);
  // 주어진 size 크기 배열의 통계 정보 (최소값, 최대값, 평균, 분산, 표준편차)를 산출.
  // 계산된 통계 정보는 화면으로 출력
- void fgetArrayStatistics(FILE *fout, int *array, int size);
  // 주어진 size 크기배열의 통계 정보 (최소값, 최대값, 평균, 분산, 표준편차)를 산출.
  // 계산된 통계 정보는 지정된 파일로 출력
- void genBigRandArray(int *array, int size);
  // 주어진 size (50000이상)개수의 중복되지 않는 난수 (random number)를 생성하여
  // 주어진 (포인터로 지정된) 배열에 저장

3) Data_array.c

- 다음과 같은 data_array[]를 선언하며 0보다 큰 정수 데이터를 제공.
- 데이터 원소의 마지막은 -1로 표시

```
/* Data_array.c */
#include "BigArray.h"
```

```
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) 데이터 파일 입력

- Data_input.txt 파일에 0보다 큰 정수 데이터를 나열하고, 마지막에 -1로 표시할 것

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

- 이 데이터 파일을 입력하여 data_array[] 배열에 저장하는 기능을 구현하고, 이를 main.c 파일에 포함되어 있는 관련 함수에서 사용 할 것

2) 파일 출력

- Data_output.txt 파일에 배열의 통계 처리 결과를 출력하는 기능을 구현하고, 이를 main.c 파일에 포함되어 있는 관련 함수에서 사용 할 것

5.3 동적 배열 생성 및 중복되지 않는 난수 생성

1) 동적 배열 생성

- 표준 입력장치로부터 배열 크기 (50000 이상)를 입력받아, 동적 배열을 생성
- 동적 배열 사용 후에는 free() 함수를 사용하여 동적 메모리 반환

2) 중복되지 않는 난수 배열

- void genBigRandArray(int *array, int size) 함수에서 중복되지 않는 난수를 생성한 후, 주어진 size 크기의 배열 array에 차례로 저장

5.4 main.c 파일

```
/* main.cpp */
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include "BigArray.h"

#define ESC 0x1B

//#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"
#define Data_Output_File "Data_output.txt"

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

```
        FILE *fout;
        char menu;

        fout = fopen(Data_Output_File, "w");
        if (fout == NULL)
        {
                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);
                        break;
                case '2':
                        arrayStatistics_externArray(fout);
                        break;
                case '3':
                        arrayStatistics_fileDataArray(fout);
                        break;
                case '4':
                        arrayStatistics_inputArray(fout);
                        break;
                case '5':
                        arrayStatistics_genBigRandArray(fout);
                        break;

                default:
                        break;
                }
        }
        fclose(fout);
        return 0;
}

void arrayStatistics_basicArray(FILE* fout)
{
        int num_data = 10;
        //int data_array[MAX_NUM_DATA] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
        int data_array[MAX_NUM_DATA] = { 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 };

        printf("\nArrayStatistics_basicArray .....\n");
        fprintf(fout, "\nArrayStatistics_basicArray .....\n");
        getArrayStatistics(data_array, num_data);
        fgetArrayStatistics(fout, data_array, num_data);
        printf("arrayStatistics_basicArray - completed. Result is also stored in output
          file(%s).\n", Data_Output_File);
}

void arrayStatistics_externArray(FILE* fout)
```

```
{
        int num_data = 0;

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

        // 외부 파일에 선언되어 있는 data_array[]의 데이터 원소들을 차례로
        // 읽고 -1이 아닌 원소의 개수를 파악하여 num_data로 설정할 것.

        getArrayStatistics(data_array, num_data);
        fgetArrayStatistics(fout, data_array, num_data);
        printf("arrayStatistics_basicArray - completed. Result is also stored in output
          file(%s).\n", Data_Output_File);
}

void arrayStatistics_fileDataArray(FILE *fout)
{
        . . . .

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

        // 데이터 파일 입력 기능을 이곳에 구현할 것
        getArrayStatistics(data_array, num_data);
        fgetArrayStatistics(fout, data_array, num_data);
        printf("arrayStatistics_fileDataArray - completed. Result is also stored in output
file
          (%s).\n", Data_Output_File);
}

void arrayStatistics_inputArray(FILE* fout)
{
        . . . .

        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);

        // 표준 입력장치 (keyboard)를 사용한 데이터 입력을 이곳에 구현할 것.
        getArrayStatistics(data_array, num_data);
        fgetArrayStatistics(fout, data_array, num_data);
        printf("arrayStatistics_inputArray - completed. Result is also stored in output file
          (%s).\n", Data_Output_File);
}

void arrayStatistics_genBigRandArray(FILE* fout)
{
        . . . .

        printf("\nArrayStatistics_genBigRandArray .....\n");
        fprintf(fout, "\nArrayStatistics_genBigRandArray .....\n");
        printf("Size of big array (bigger than 50000) = ");
        scanf("%d", &size);

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

```
        getArrayStatistics(array, size);
        fgetArrayStatistics(fout, array, size);
        // 동적 배열의 반환 기능을 이곳에 구현할 것.
        printf("arrayStatistics_genBigRandArray - completed. Result is also stored in
           output file (%s).\n", Data_Output_File);
}
```

5.5 실행결과

1) 화면 출력 결과

```
Test Array Handling (1: data_array; 2: extern_array; 3: data_file; 4: data_input; 5: gen_BigRandArray; Esc: terminate) : 1
ArrayStatistics_basicArray .....
Total ( 10) integer data :
     11      12      13      14      15      16      17      18      19      20

min ( 11), max ( 20), sum (  155.00), average (   15.50), variance (    8.25), standard deviation (    2.87)
arrayStatistics_basicArray - completed. Result is also stored in output file(Data_output.txt).

Test Array Handling (1: data_array; 2: extern_array; 3: data_file; 4: data_input; 5: gen_BigRandArray; Esc: terminate) : 2
ArrayStatistics_externArray .....
Total ( 35) integer data :
      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
min (  1), max ( 35), sum (  630.00), average (   18.00), variance (  102.00), standard deviation (   10.10)
arrayStatistics_basicArray - completed. Result is also stored in output file(Data_output.txt).

Test Array Handling (1: data_array; 2: extern_array; 3: data_file; 4: data_input; 5: gen_BigRandArray; Esc: terminate) : 3
ArrayStatistics_fileDataArray .....
Total ( 20) integer data :
      1       2       3       4       5       6       7       8       9      10
     11      12      13      14      15      16      17      18      19      20

min (  1), max ( 20), sum (  210.00), average (   10.50), variance (   33.25), standard deviation (    5.77)
arrayStatistics_fileDataArray - completed. Result is also stored in output file (Data_output.txt).

Test Array Handling (1: data_array; 2: extern_array; 3: data_file; 4: data_input; 5: gen_BigRandArray; Esc: terminate) : 4
ArrayStatistics_inputArray .....
Please input the number of integers (less than 100) = 10
Input 10  integer data : 10 9 8 7 6 5 4 3 2 1
Total ( 10) integer data :
     10       9       8       7       6       5       4       3       2       1

min (  1), max ( 10), sum (   55.00), average (    5.50), variance (    8.25), standard deviation (    2.87)
arrayStatistics_inputArray - completed. Result is also stored in output file (Data_output.txt).

Test Array Handling (1: data_array; 2: extern_array; 3: data_file; 4: data_input; 5: gen_BigRandArray; Esc: terminate) : 5
ArrayStatistics_genBigRandArray .....
Big Array Size (more than 50000) = 1000000
Creating big random integer array (size : 1000000)
Total (1000000) integer data :
  43772  992633  524109  333728  118867  451200  582276  428751   87061  962571
 429215  972163  970055  199705  125982  744326  967430  434078  807127  314710

  . . . .
 630262  864460  689656  472410  918082  875151  482396  100257  780860  896350
  19128  720252  294787  294308  923840  849253  925318  466734  520854  890764

min (  0), max (999999), sum (499999500000.00), average (499999.50), variance (83333333333.79), standard deviation (288675.
arrayStatistics_genBigRandArray - completed. Result is also stored in output file (Data_output.txt).
```

2) 파일 출력 결과

```
ArrayStatistics_basicArray .....
Total ( 10) integer data :
       11      12      13      14      15      16      17      18      19      20

min ( 11), max ( 20), sum (  155.00), average (   15.50), variance (    8.25), standard deviation (    2.87)

ArrayStatistics_externArray .....
Total ( 35) integer data :
        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
min (  1), max ( 35), sum (  630.00), average (   18.00), variance (  102.00), standard deviation (   10.10)

ArrayStatistics_fileDataArray .....
Total ( 20) integer data :
        1       2       3       4       5       6       7       8       9      10
       11      12      13      14      15      16      17      18      19      20

min (  1), max ( 20), sum (  210.00), average (   10.50), variance (   33.25), standard deviation (    5.77)

ArrayStatistics_inputArray .....
Total ( 10) integer data :
       10       9       8       7       6       5       4       3       2       1

min (  1), max ( 10), sum (   55.00), average (    5.50), variance (    8.25), standard deviation (    2.87)

ArrayStatistics_genBigRandArray .....
Total (1000000) integer data :
    43772  992633  524109  333728  118867  451200  582276  428751   87061  962571
 429215  972163  970055  199705  125982  744326  967430  434078  807127  314710

   . . . .
 638262  864460  689656  472410  918082  875151  482396  100257  780860  896350
  19128  720252  294787  294308  923840  849253  925318  466734  520854  890764

min (  0), max (999999), sum (499999500000.00), average (499999.50), variance (83333333333.79), standard deviation (288675.
```

<Oral Test>

Q5.1 다중 소스 파일 구조의 프로그램 개발에서 사용자 정의 헤더 파일에 포함되는 내용에 대하여 설명하라.

(KeyPoints: 다수의 파일에서 공통적으로 사용되어야 하는 정보를 고려할 것.)

Q5.2 전처리기 지시어를 사용하여 헤더 파일의 중복 포함을 방지하는 방법과 조건부 컴파일을 실행하는 방법을 각각 예를 들어 설명하라.

(KeyPoints: #ifndef - #endif, #ifdef - #endif 를 사용하여 각각 설명할 것.)

Q5.3 RAND_MAX (32767) 보다 큰 정수인 BIG_RAND_MAX (2^{30} - 1) 까지의 난수를 중복되지 않게 생성하여 주어진 배열에 차례로 담아주는 genBigRandArray (int *array, int size)의 동작 절차에 대하여 상세하게 설명하라.

(KeyPoints: 30-비트 크기의 난수를 생성하는 절차, 생성된 난수가 이미 생성된 적이 있는가를 검사하는 방법을 설명할 것.)

Q5.4 주어진 배열의 통계적 특성을 산출하기 위하여 평균값 (mean), 분산 (variance), 표준편차 (standard deviation)를 계산하는 방법에 대하여 상세하게 설명하라.

(KeyPoints: 평균, 분산, 표준편차 계산 방법을 설명할 것.)