**Homework 5**

|  |
| --- |
| 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 | 2021.04.05.월 |

1. **다중 소스파일 프로그램의 장점에 대하여 설명하라.**

단일 소스 파일로 작성하는 경우에는 파일의 크기가 너무 커지고 소스 파일을 다시 사용하기가 어렵다. 하지만, 다중 소스 파일로 작성하는 경우에는 서로 관련된 코드만을 모아서 하나의 소스 파일로 할 수 있고 소스 파일을 재사용하기가 간편하다.

1. **정수형 난수 배열 관련 함수들을 위한 BigArray.cpp, BigArray.h 구현**

* rand() 함수를 사용하여 중복되지 않는 정수형 난수 배열을 생성하는 함수 void genBigRandArray(int \*array, int size)를 BigArray.cpp 소스코드 파일에 구현하라. size값은 50000보다 큰 숫자가 되도록 하며, 생성되는 난수 배열은 0 ~ size – 1의 값을 가지며, 중복되지 않아야 한다.

// 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

}

* 주어진 배열에 포함되어 있는 정수 원소들의 위치를 뒤섞는 void suffleBigArray(int \*array, int size)를 BigArray.cpp소스코드 파일에 구현하라.

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

int i1, i2, d;

srand((unsigned)time(NULL));

for (int i = 0; i < size / 2; i++) {

i1 = (((unsigned long)rand() << 15 | rand())) % size;

i2 = (((unsigned long)rand() << 15 | rand())) % size;

// suffle

d = array[i1];

array[i1] = array[i2];

array[i2] = d;

}

}

* 정수 배열의 내용을 화면으로 출력하는 함수 printBigArraySample(int \*array, int size, int line\_size = 10, int num\_sample\_lines = 2)를 BigArray.cpp파일에 구현하라.

// 실행창에 배열 정보 출력

void printBigArraySample(int\* array, int size, int line\_size = 10, int num\_sample\_lines = 2) {

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");

}

}

* 정수 배열이 내용을 지정된 파일에 출력하는 함수 fprintBigArraySample(FILE\* fout, int \*array, int size, int line\_size = 10, int num\_sample\_lines = 2)를 BigArray.cpp에 구현하라.

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");

}

}

// 지정된 파일에 배열 정보 출력

void fprintBigArraySample(FILE\* fout, int\* array, int size, int line\_size = 10, int num\_sample\_lines = 2) {

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;

}

}

* 위에서 구현한 배열 관련 함수들의 함수 원형을 포함하는 헤더파일 BigArray.h를 구현하라.

#ifndef BIG\_ARRAY\_H // 헤더 중복 방지

#define BIG\_ARRAY\_H

#define Data\_Input\_File "Data\_input.txt"

#define Data\_Output\_File "Data\_output.txt"

#include <stdio.h>

#include <math.h>

#include <time.h>

void printBigArraySample(int\* array, int size, int line\_size = 10, int num\_sample\_lines = 2);

void fprintBigArraySample(FILE\* fout, int\* array, int size, int line\_size = 10, int num\_sample\_lines = 2);

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);

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

#endif

* main()함수에서는 BigArray.h 헤더 파일을 포함시켜 정수형 배열 관련 함수를 사용할 수 있도록 구성하라.

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <math.h>

#include <time.h>

#include "BigArray.h"

* main() 함수에서 정수형 배열을 선언하고, genBigRandArray() 함수를 호출하여 이 정수형 배열에 중복되지 않는 난수를 생성하고 fprintBigArraySample()함수를 사용하여 출력 파일 “Output.txt”에 출력하라.

텍스트, 점수판, 스크린샷, 닫기이(가) 표시된 사진

자동 생성된 설명

* 전체 프로그램을 다중 소스파일로 구현하라.



* 출력 결과

텍스트이(가) 표시된 사진

자동 생성된 설명

* 동적 메모리 할당을 사용하여 100만개 이상의 정수 배열을 생성하고, 중복되지 않는 난수를 생성하라.
* 출력결과

텍스트이(가) 표시된 사진

자동 생성된 설명