프로그래밍언어 (실습)

# 실습 5 – (보충설명) 전처리기, 파일 입출력, 동적 배열 생성, 다중 파일 프로그램


**교수 김 영 탁, 황현동**

**영남대학교 기계IT대학 정보통신공학과**

(Tel : +82-53-810-2497; Fax : +82-53-810-4742

http://antl.yu.ac.kr/; E-mail : ytkim@yu.ac.kr)

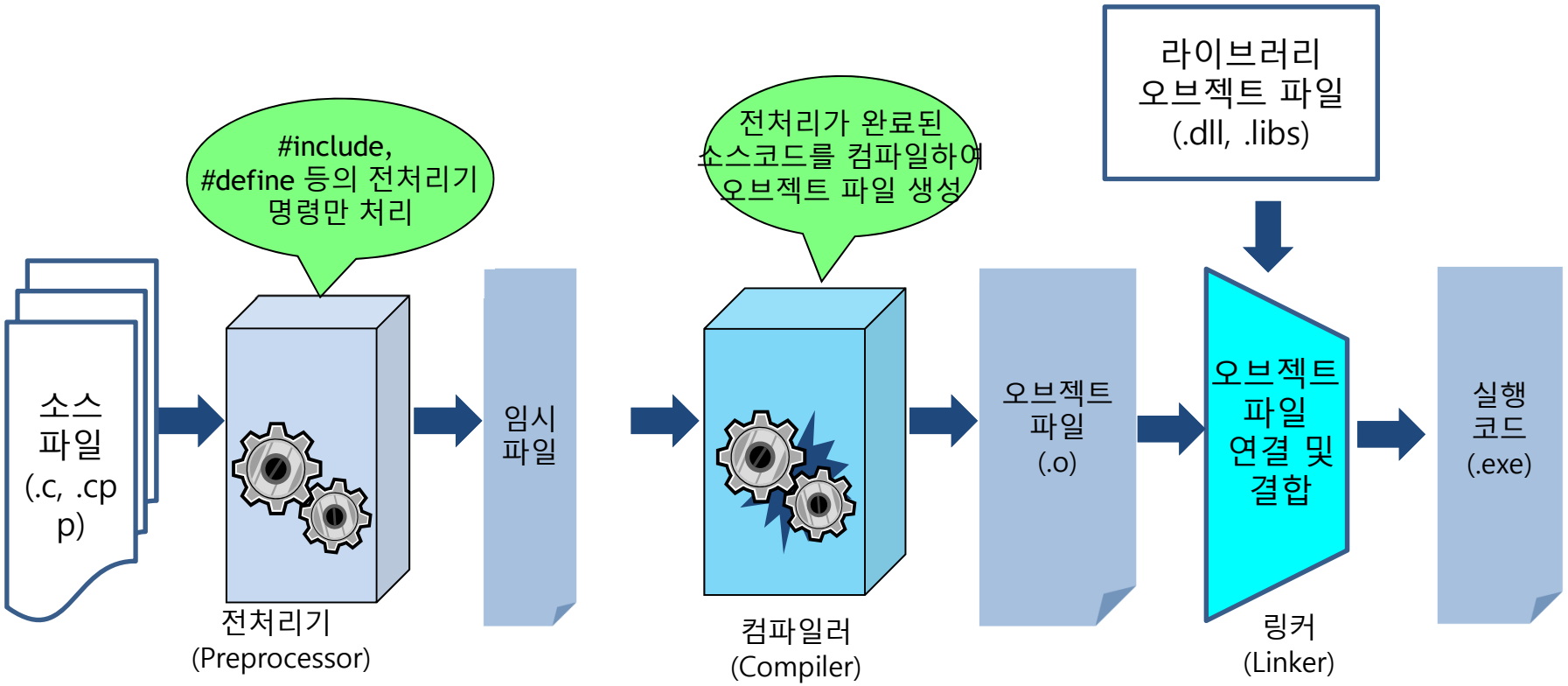
# Outline

- 전처리기
- 파일입출력
  - fopen(), fclose()
  - fscanf(), fprintf()
  - fgets()
- 배열
  - 배열의 선언
  - 다양한 배열의 준비 방법
  - 동적 배열의 생성
- 실습 5
  - 다중파일 프로그램 구조의 BigRandArray 생성 및 사용

# 전처리기

# 전처리기란?

◆ *전처리기 (preprocessor)*는 컴파일하기에 앞서서 소스 파일을 처리하는 컴파일러의 한 부분



# 전처리기 (preprocessor) 지시어

| 전처리기 지시어 (예) | 의미 |
|---|---|
| #include <stdio.h><br>#include "MyHeaderFile.h" | 지정된 파일 (헤더파일)을 포함 |
| #define PI 3.141592<br>#define SQUARE(x) ( (x) * (x) )<br>#define MAX(x, y)<br>( ( (x) >= (y) ) ? (x) : (y) ) | 기호 상수의 값을 지정<br>매크로 함수 지정 |
| #undef | 이전에 정의되었던 매크로 정의를 해제 |
| #if | 지정된 조건이 참일 경우 #else나 #endif 까지의 구간을 실행 |
| #else | #if에서 지정된 조건이 참이 아닐 경우 #endif 까지의 구간을 실행 |
| #endif | #if, #else, #ifdef, #ifndef, #elif 등의 전처리 지시어 조건의 구역을 끝을 지정 |
| #ifdef DEBUG | 해당 기호상수가 지정되어 있으면,<br>#endif가 나타날 때까지의 구간을 실행 |
| #ifndef | 해당 기호상수가 지정되어 있지 않으면,<br>#endif가 나타날 때까지의 구간을 실행 |
| #line | 행 번호를 출력 |
| #elif | else-if를 의미 |
| #pragma | 시스템에 따라 다른 의미를 부여 |

# <limits.h> 헤더파일에서 정의하는 기호상수

| 기호상수 | 값 | 의미 |
|---|---|---|
| CHAR_BIT | 8 | 문자의 비트수 (비트단위로 나누어지지 않는 최소 크기) |
| SCHAR_MIN | -128 | 부호있는 문자형의 최소값 |
| SCHAR_MAX | 127 | 부호있는 문자형의 최대값 |
| UCHAR_MAX | 255 (0xFF) | 부호없는 문자형의 최대값 |
| CHAR_MIN | -128<br>0 if /j option used | (부호있는) 문자형의 최소값; /j 옵션이 사용되면 0 |
| CHAR_MAX | 127<br>255 if /j option used | (부호있는) 문자형의 최대값; /j 옵션이 사용되면 255 |
| SHRT_MIN | -32768 | 부호있는 short 형의 최소값 |
| SHRT_MAX | 32767 | 부호있는 short 형의 최대값 |
| USHRT_MAX | 65535 (0xFFFF) | 부호없는 short 형의 최대값 |
| INT_MIN | -2147483648 | 부호있는 정수 (int) 형의 최소값 |
| INT_MAX | 2147483647 | 부호있는 정수 (int) 형의 최대값 |
| UINT_MAX | 4294967295<br>(0xFFFFFFFF) | 부호없는 정수 (unsigned int) 형의 최대값 |
| LONG_MIN | -2147483648 | (부호있는) Long 형 정수의 최소값 |
| LONG_MAX | 2147483647 | (부호있는) Long 형 정수의 최대값 |
| ULONG_MAX | 4294967295<br>(0xFFFFFFFF) | 부호없는 Long 형 정수의 최대값 |
| _I64_MIN | -9223372036854775808 | __int64형 (64비트) 정수의 최소값 |
| _I64_MAX | 9223372036854775807 | __int64형 (64비트) 정수의 최대값 |
| _UI64_MAX | 18446744073709551615<br>(0xFFFFFFFFFFFFFFFF) | 부호없는 __int64형 (64비트) 정수의 최대값 |

# 내장 매크로

## ◆ 내장 매크로: 미리 정의된 매크로

| 내장 매크로 | 설명 |
|---|---|
| __DATE__ | 이 매크로를 만나면 현재의 날짜(월 일 년)로 치환된다. |
| __TIME__ | 이 매크로를 만나면 현재의 시간(시:분:초)으로 치환된다. |
| __LINE__ | 이 매크로를 만나면 소스 파일에서의 현재의 라인 번호로 치환된다. |
| __FILE__ | 이 매크로를 만나면 소스 파일 이름으로 치환된다. |

- printf("컴파일 날짜=%s₩n", __DATE__);
- printf("치명적 에러 발생 파일 이름=%s 라인 번호= %d₩n", __FILE__, __LINE__);

# 비트 관련 매크로

- 매크로들은 변수를 받아서 특정 비트값을 반환하거나 설정한다.

- GET_BIT(w, k)는 변수 w에서 k번째 비트의 값을 0 또는 1로 반환한다.

```
#define GET_BIT(w, k) (((w) >> (k)) & 0x01)
```

- SET_BIT_ON(w, k)는 변수 w의 k번째 비트를 1로 설정하는 매크로이다.

```
#define SET_BIT_ON(w, k) ((w) |= (0x01 << (k)))
```

- SET_BIT_OFF(w, k)는 변수 w의 k번째 비트를 0로 설정하는 매크로이다.

```
#define SET_BIT_OFF(w, k) ((w) &= ~(0x01 << (k)))
```

# 파일 입출력

# File open (1)

## ◆ File open의 의미

- 파일에서 데이터를 읽거나 쓸 수 있도록 모든 준비를 마치는 것을 의미
- 파일을 연 다음에는 데이터를 읽기, 쓰기 가능
- File open → File read & write → File close 순으로 진행

- FILE 구조체를 통하여 파일에 접근
  - FILE 구조체를 가리키는 포인터를 파일 포인터 (file pointer)라고 한다
  - 각각의 파일마다 하나의 파일 포인터가 필요

```
FILE *fopen (const char *name, const char *mode);
```

name : 파일의 이름을 나타내는 문자열
mode : 파일을 여는 방식

# File open (2)

## ◆ File mode

| 모드 | 설명 |
|---|---|
| "r" | 읽기 모드로 파일을 연다. |
| "w" | 쓰기 모드로 파일을 생성한다. 만약 파일이 존재하지 않으면 파일이 생성된다. 파일이 이미 존재하면 기존의 내용이 지워진다. |
| "a" | 추가 모드로 파일을 연다. 만약 똑같은 이름의 기존의 파일이 있으면 데이터가 파일의 끝에 추가된다. 파일이 없으면 새로운 파일이 만들어진다. |
| "r+" | 읽기와 쓰기 모드로 파일을 연다. 파일이 반드시 존재해야 한다. |
| "w+" | 읽기와 쓰기 모드로 파일을 생성한다. 만약 파일이 존재하지 않으면 파일이 생성된다. 파일이 존재하면 새 데이터가 기존 파일의 데이터에 덮어 쓰인다. |
| "a+" | 읽기와 추가 모드로 파일을 연다. 만약 똑같은 이름의 기존의 파일이 있으면 데이터가 파일의 끝에 추가된다. 읽기는 어떤 위치에서나 가능하다. 파일이 없으면 새로운 파일을 만든다. |
| "b" | 이진 파일 모드로 파일을 연다. |

# File close

## ◆ fclose()

- 열린 파일을 닫는 함수
- stdio.h에 정의
- 성공적으로 파일을 닫는 경우에는 0이 반환
- 만약 실패한 경우에는 -1이 반환

```
int fclose (FILE *stream);
```

# File open/close

## ◆ Sample.txt 생성

- Project -> sample.txt 생성

## ◆ File open/close 예제

```
#include <stdio.h>

int main()
{
    FILE *fp = NULL;          //FILE 포인터fp를 생성하고NULL로 초기화
    fp = fopen("sample.txt", "w");     //파일을 쓰기모드로 열고, 그 주소를 fp에 저장

    if (fp == NULL)
        printf("파일열기실패₩n");
    else
        printf("파일열기성공₩n");

    fclose(fp);               //파일 닫기

    return 0;
}
```

# fprintf()

## ◆ 형식화된 입출력

- 정수나 실수 데이터를 파일에 문자열로 바꾸어서 저장
- fprintf()
  - 사용방법은 printf()와 비슷하나, 화면이 아닌 파일에 출력

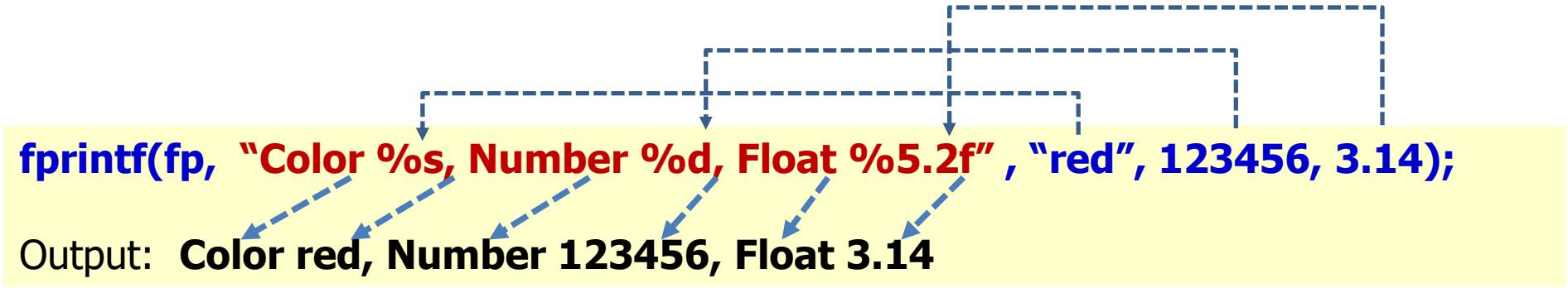
```
int fprintf( FILE *fp, const char *format, ...);
```

  - fprintf() 사용 예제

```
#include <stdio.h>
int main()
{
    int i = 23;
    float f = 1.2345;
    FILE *fp = NULL;          //FILE 포인터fp를 생성하고NULL로 초기화
    fp = fopen("sample.txt", "w");     //파일을 쓰기모드로 열고, 그 주소를 fp에 저장

    if(fp != NULL)
        fprintf(fp, "%10d %16.4f", i, f);
    fclose(fp);               //파일닫기
    return 0;
}
```

# Formatted file output in C - fprintf()

## ◆ Example




```
int x = 7;
double d = 12.5;
fprintf(fp, "%4d", x); // prints out integer x in 4 places
fprintf(fp, "%2d", 3); // prints out " 3"
fprintf(fp, "%02d", 3); // prints out "03".
fprintf(fp, "%4d, %4x", 15, 15);  // prints out "15,  F"
fprintf(fp, "%7.2f", d); // printout double d in total 7 places
                         // with 2 digits to the right of decimal points
```

# fprintf() format specifier

| Format Character | Output data type | Output |
|---|---|---|
| %d | int | signed decimal integer |
| %u | unsigned int | unsigned decimal integer |
| %o | unsigned int | unsigned octal integer |
| %x, %X | unsigned int | unsigned hexadecimal integer |
| %f, %lf | float, double | floating point numbers in decimal format |
| %e, %E | float, double | floating point numbers in scientific format (e.g., 1.2345e-001 or 1.0E-20) |
| %g, %G | float, double | selects %f or %e according to the value |
| %c | char | character |
| %s | char * | string indicated by a character pointer |
| %p | void * | address value of the pointer |
| %n | int * | address value of the pointer |

# fprintf()에서의 출력공간 및 정렬 지정

| 포맷 문자 (Format Character) | 출력 데이터 유형 (Output data type) | 출력 포맷 (Output) |
|---|---|---|
| %8d | int | 10진수를 8칸에 오른쪽 맞춤으로 출력 |
| %8o | unsigned int | 8진수를 8칸에 오른쪽 맞춤으로 출력 |
| %8x, %8X | unsigned int | 16진수를 8칸에 오른쪽 맞춤으로 출력 |
| %#d | int | 10진수를 출력 (10진수의 경우 별도의 prefix없음) |
| %#o | unsigned int | 8진수를 prefix (0)과 함께 출력 |
| %#x, %#X | unsigned int | 16진수를 prefix (0x 또는 0X)과 함께 출력 |
| %#08d | int | 10진수를 8칸에 오른쪽 맞춤으로 출력하며, 앞의 빈자리에는 0을 채워줌 (10진수의 경우 별도의 prefix없음) |
| %#08o | unsigned int | 8진수를 prefix (0)과 함께 8칸에 오른쪽 맞춤으로 출력하며, 앞의 빈자리에는 0을 채워줌 |
| %#08x, %#08X | unsigned int | 16진수를 prefix (0x 또는 0X)과 함께 8칸에 오른쪽 맞춤으로 출력하며, 앞의 빈자리에는 0을 채워줌 |
| %-8d | float, double | 10진수를 8칸에 왼쪽 맞춤으로 출력 |
| %+8d | float, double | 10진수를 8칸에 오른쪽 맞춤으로 + 부호와 함께 출력 |
| %20s | char * | 문자열을 20칸에 오른쪽 맞춤으로 출력 |
| %-20s | char * | 문자열을 20칸에 왼쪽 맞춤으로 출력 |

# fscanf()

- fscanf()
  - scanf()와 사용법은 비슷하지만 입력 대상이 키보드가 아닌 파일

```
int fscanf( FILE *fp, const char *format, ...);
```

  - fscanf() 사용 예제

```
#include <stdio.h>

int main()
{
    int i;
    float f;
    FILE *fp = NULL;          //FILE 포인터fp를 생성하고 NULL로 초기화
    fp = fopen("sample.txt", "r");    //파일을 읽기모드로 열고, 그 주소를 fp에 저장

    if(fp != NULL)
        fscanf(fp, "%d %f", &i, &f);
    printf(" %d₩n %f₩n", i, f);    //화면에 출력
    fclose(fp);               //파일 닫기

    return 0;
}
```

# fgets(), token()

## ◆ fgets()

- The C library function **char *fgets(char *str, int n, FILE *stream)** reads a line from the specified stream and stores it into the string pointed to by **str**.
- It stops when either **(n-1)** characters are read, the newline character is read, or the end-of-file (EOF) is reached, whichever comes first.

## ◆ token(), strtok()

- char *token;
- char sepChars[] = ":, ₩t₩n;";
- char str[MAX_STRING_LEN] = “This is a test string.”;
- token = strtok(str, sepChars);
- token = strtok(NULL, sepChars);

# Sample program with fgets() and token()

```
/* Test program for file input and output (1) */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include<Windows.h>
#define MAX_STRING_LEN 256
int main()
{
	FILE *fp_in, *fp_out;
	char *token;
	char sepChars[] = ":, \t\n;";
	char str[MAX_STRING_LEN] = {'\0'};
	int line_count = 1;

	fp_in = fopen("TestInput.txt", "r");
	if (fp_in == NULL)
	{
		printf("Error in file open - TestInput.txt !!\n");
		exit(-1);
	}

	fp_out = fopen("TestOutput.txt", "w");
	if (fp_out == NULL)
	{
		printf("Error in file open - TestOutput.txt !!\n");
		exit(-1);
	}
```

# Sample program with fgets() and token()

```
/* Test program for file input and output  (2) */

        printf("Testing fgets() and strtok() ...\n");
        while (fgets(str, MAX_STRING_LEN, fp_in) != NULL)
        {
                printf("Line %2d : %s", line_count++, str);
                fprintf(fp_out, "%s", str);
                token = strtok(str, sepChars);
                int i = 1;
                printf("\t");
                fprintf(fp_out, "\t");
                while (token != NULL)
                {
                        printf("(Token %d) %s ", i++, token);
                        fprintf(fp_out, "(Token %d) %s ", i, token);
                        token = strtok(NULL, sepChars);
                }
                printf("\n");
                fprintf(fp_out, "\n");
        }

        fclose(fp_out);
        printf("\n");
}
```

# Test Results

## ◆ Input file

TestOutput.txt | TestInput.txt | ThesaurusDict.cpp

```
1  Test input file.
2  The second line.
3  The third line with numbers : 0123456789.
4
```

## ◆ Output file

TestOutput.txt | TestInput.txt | ThesaurusDict.cpp

```
1  Test input file.
2      (Token 2) Test (Token 3) input (Token 4) file.
3  The second line.
4      (Token 2) The (Token 3) second (Token 4) line.
5  The third line with numbers : 0123456789.
6      (Token 2) The (Token 3) third (Token 4) line (Token 5) with (Token 6) numbers (Token 7) 0123456789.
7
```

# fprintArray()

```
void fprintArray(FILE *fout, int *array, int size, int line_size)
{
    for (int i = 0, count = 0; i < size; i++)
    {
        fprintf(fout, "%5d ", array[i]);
        count++;
        if (count % line_size == 0)
           fprintf(fout, "\n");
    }
    fprintf(fout, "\n");
}
```

```
/* Array.c (4) */

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;
        if (data > max)
           max = data;
    }
    avg = sum / (double) size;
    sq_diff_sum = 0.0;
    for (int i = 0; i < size; i++)
    {
        diff = array[i] - avg;
        sq_diff_sum += diff * diff;
    }
    var = sq_diff_sum / (double)size;
    std_dev = sqrt(var);
    fprintf(fout, "Total (%3d) integer data : \n  ", size);
    fprintArray(fout, array, size, 10);
    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);
}
```

# 파일 입력, 출력 예제 프로그램 – 데이터 배열

```
/* main.cpp (4) */

void arrayStatistics_fileDataArray(FILE *fout)
{
    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_List.txt", "r");
    if (fin == NULL)
    {
        printf("Error in opening input data file !!\n");
        return;
    }
    while (fscanf(fin, "%d", &data) != EOF)
    {
        if (data == -1)
            break;
        data_array[num_data] = data;
        num_data++;
    }
    fclose(fin);
    getArrayStatistics(data_array, num_data);
    fgetArrayStatistics(fout, data_array, num_data);
    printf("arrayStatistics_fileDataArray - completed. Result is also stored in output file.\n");
}
```

# 파일 입출력 응용 프로그램 실행 결과

## ◆ 입력 데이터 파일 (DataList.txt)

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

## ◆ 출력 결과 파일

```
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_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 (  5) integer data :
     3     7     9    13    17
min (  3), max ( 17), sum (   49.00), average (    9.80), variance (   23.36), standard deviation (    4.83)
```

# 다중 소스 파일

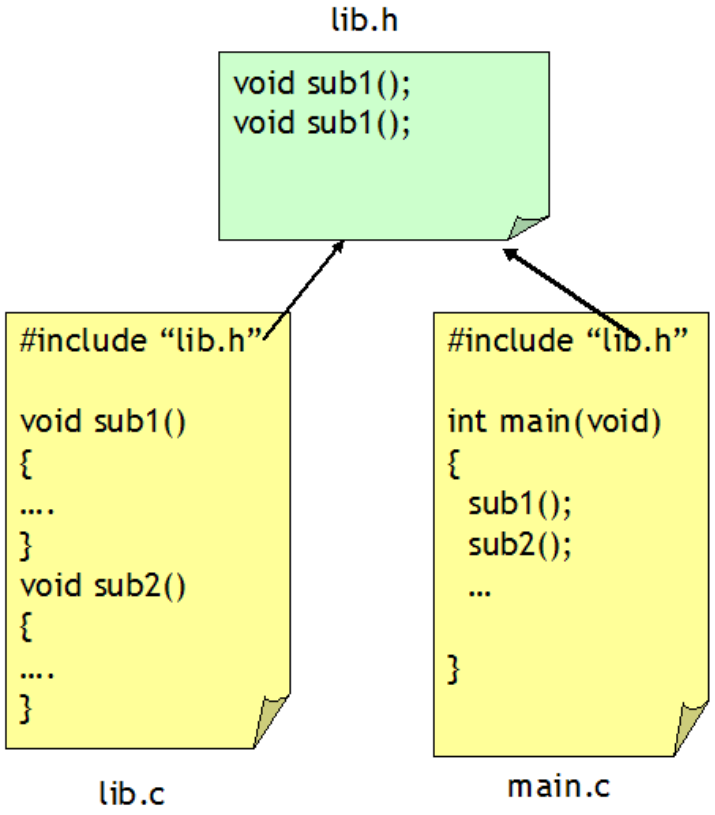
# 다중 소스 파일

## ◆ 단일 소스 파일

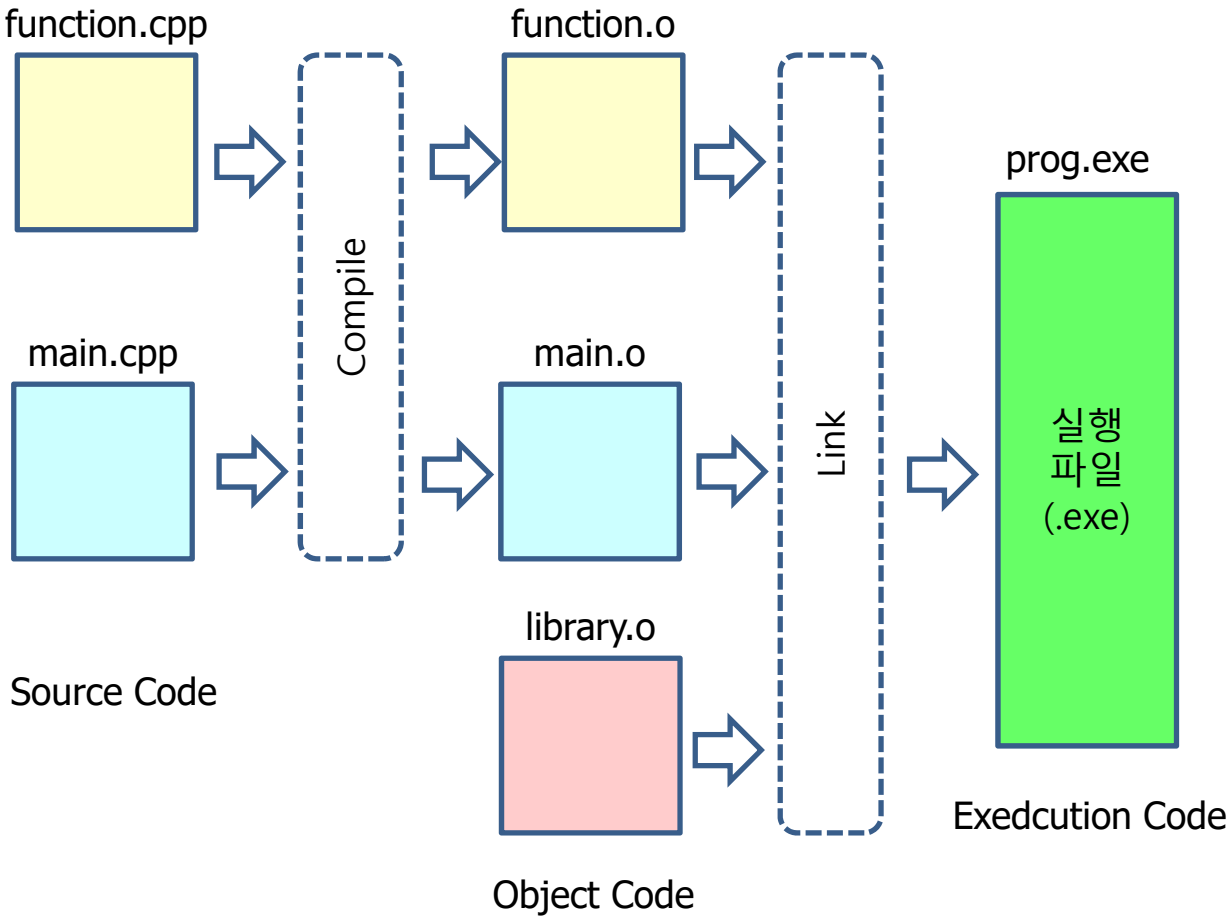
- 파일의 크기가 너무 커진다.
- 소스 파일을 다시 사용하기가 어려움

## ◆ 다중 소스 파일

- 서로 관련된 코드만을 모아서 하나의 소스 파일로 할 수 있음
- 소스 파일을 재사용하기가 간편함



# 다중 소스 파일의 컴파일과 링크



# 예제

*multiple_source.c*

```c
// 다중 소스 파일
#include <stdio.h>
#include "power.h"

int main(void)
{
    int x,y;

    printf("x의 값을 입력하시오:");
    scanf("%d", &x);
    printf("y의 값을 입력하시오:");
    scanf("%d", &y);
    printf("%d의 %d 제곱값은 %f\n", x, y, power(x, y));

    return 0;
}
```

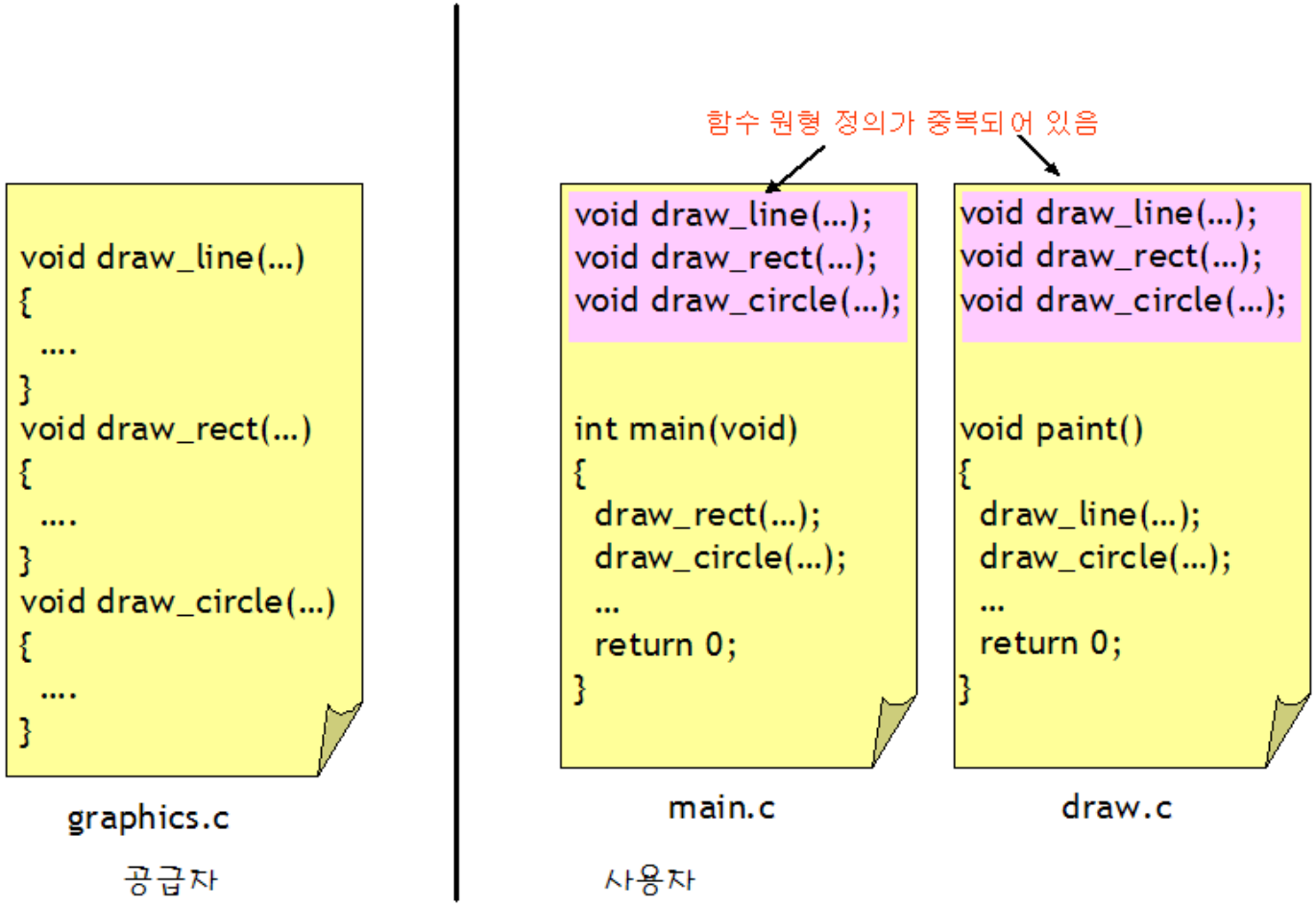
*power.h*

```c
// power.c에 대한 헤더 파일
#ifndef POWER_H
#define POWER_H

double power(int x, int y);
#endif
```

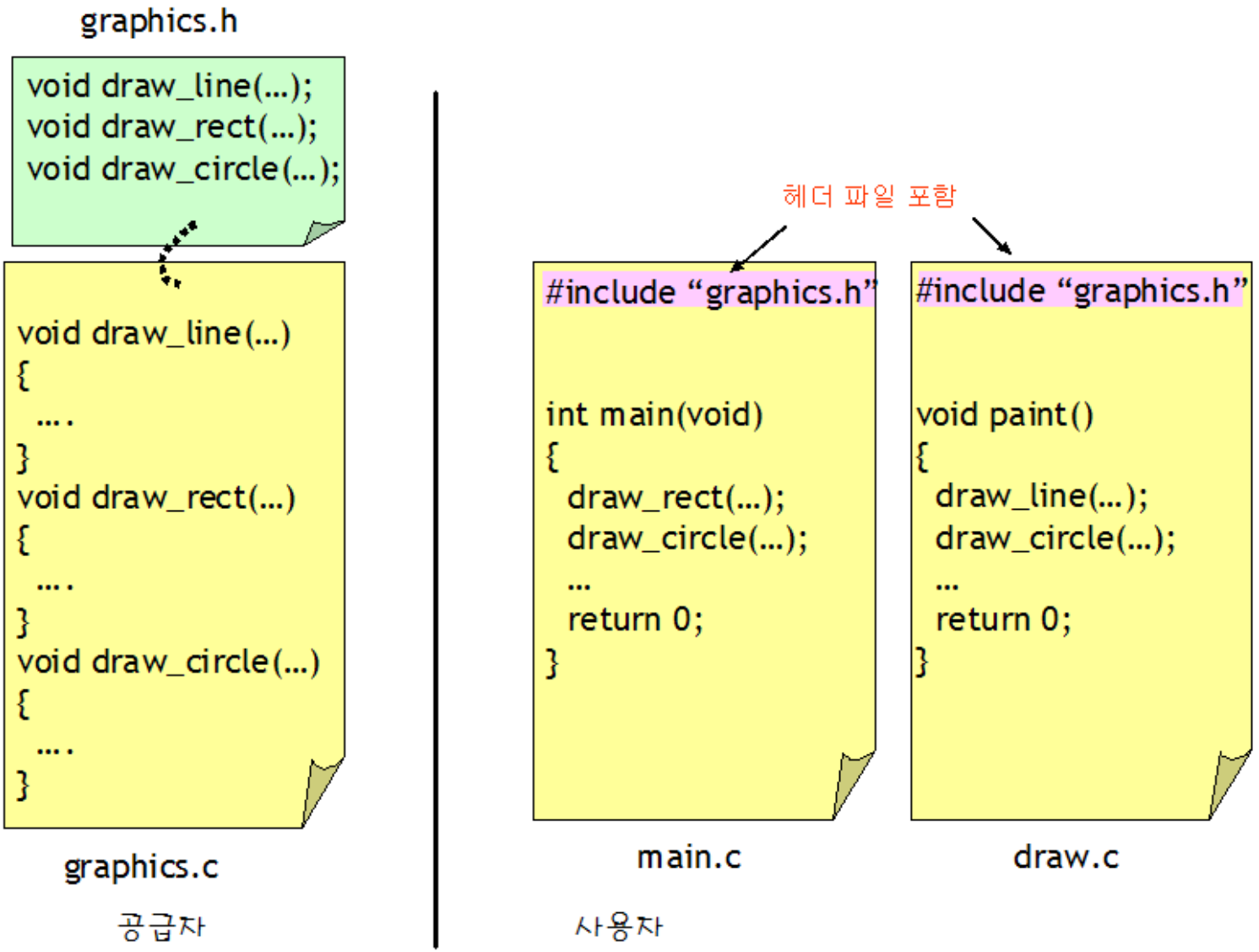
*power.c*

```c
// 다중 소스 파일
#include "power.h"
double power(int x, int y)
{
    double result = 1.0;
    int i;

    for(i = 0;i < y; i++)
        result *= x;

    return result;
}
```

# 헤더 파일을 사용하지 않으면



# 헤더 파일을 사용하면


graphics.h
void draw_line(...);
void draw_rect(...);
void draw_circle(...);
void draw_line(...)
{
....
}
void draw_rect(...)
{
....
}
void draw_circle(...)
{
....
}
graphics.c
공급자
헤더 파일 포함
#include “graphics.h”
int main(void)
{
draw_rect(...);
draw_circle(...);
...
return 0;
}
main.c
#include “graphics.h”
void paint()
{
draw_line(...);
draw_circle(...);
...
return 0;
}
draw.c
사용자

# 다중 소스 파일에서 외부 변수 사용 - extern

```
/* main.c */

double gx, gy;
int main(void)
{
   gx = 123.567;
   gy = 456.789;
   . . . .
}
```

(a) 전역변수 선언

```
/* average.c */

extern double gx, gy;
double average (void)
{
   . . . .
   avg = (gx + gy) / 2.0;
}
```

(b) 외부 (extern) 변수 선언

# 헤더 파일 이중 포함 방지

```
#include <stdio.h>

#include "rect.h"
#include "rect.h"

#define DEBUG

void draw_rect(const RECT *r)
{
#ifdef DEBUG
    printf("draw_area(x=%d, y=%d, w=%d, h=%d) \n", r->x, r->y, r->w, r->h);
#endif
}
```

구조체의 정의가 이중으로 포함 되어서 오류가 발생한다.

# 헤더 파일 이중 포함 방지

```
#ifndef RECT_H
#define RECT_H

struct rect {
    int x, y, w, h;
};

typedef struct rect RECT;
void draw_rect(const RECT *);
double calc_area(const RECT *);
void move_rect(RECT *, int, int);

#endif
```

RECT_H가 정의되어 있지 않은 경우에만 포함시킨다.

RECT_H 매크로를 정의한다.

# 실습 5

# BigArray.h

```
/* BigArray.h*/

#ifndef BIG_ARRAY_H   // 헤더 파일의 중복 포함 방지
#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
```

# Data for Array

## ◆ Data_Array.c

```
/* Data_Array.c */
#include "Array.h"

int data_array[MAX_NUM_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,
    -1
};
```

## ◆ Data_File.txt

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

# main.cpp

```
/* main.cpp (1) */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include "BigArray.h"

#define ESC 0x1B

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[])
{
    int num_data, data;
    double sum = 0.0, var, diff, sq_diff_sum = 0.0, avg, std_dev;
    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;
    }
```

```
/* main.cpp (2) */

    while (1)
    {
        printf("\nTest Array Handling (1: data_array; 2: extern_array;
         3: data_file; 4: data_input; 5: gen_BigRandArray; 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;
}
```

```
/* main.cpp (3) */

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[50] = { 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);
}
```

```
/* main.cpp (4) */

void arrayStatistics_externArray(FILE* fout)
{
    int num_data = 0;
    extern int data_array[100];

    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);
}
```

```
/* main.cpp (5) */

void arrayStatistics_fileDataArray(FILE *fout)
{
    FILE *fin;
    int data, num_data = 0;
    int data_array[100] = { 0 };

    printf("\nArrayStatistics_fileDataArray .....\n");
    fprintf(fout, "\nArrayStatistics_fileDataArray .....\n");
    fin = fopen(Data_Input_File, "r");
    if (fin == NULL)
    {
        printf("Error in opening input data file !!\n");
        return;
    }
    // 데이터 파일 입력 기능을 이곳에 구현할 것
    // 데이터 파일에서 -1 (sentinel)이 입력되면 파일 입력을 종료할 것

    fclose(fin);
    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);
}
```

```
/* main.cpp (6) */

void arrayStatistics_inputArray(FILE* fout)
{
    int num_data, data;
    int data_array[100] = { 0 };
    printf("\nArrayStatistics_inputArray .....\n");
    fprintf(fout, "\nArrayStatistics_inputArray .....\n");
    printf("Please input the number of integers (less than %d) = ", 100);
    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);
}
```

```
/* main.cpp (7) */

void arrayStatistics_genBigRandArray(FILE* fout)
{
    int num_data;
    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;
    }
    genBigRandArray(dyn_array, num_data);
    getArrayStatistics(dyn_array, num_data);
    fgetArrayStatistics(fout, dyn_array, num_data);
    printf("arrayStatistics_genBigRandArray - completed. Result is also stored in output file (%s).₩n",
         Data_Output_File);
}
```

# 실행결과 - 화면 출력

```
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

 ..... .
 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 (288675.13)
arrayStatistics_genBigRandArray - completed. Result is also stored in output file (Data_output.txt).
```

# 실행결과 - 파일 출력

```
1
2   ArrayStatistics_basicArray .....
3   Total ( 10) integer data :
4          11      12      13      14      15      16      17      18      19      20
5
6   min ( 11), max ( 20), sum (  155.00), average (   15.50), variance (    8.25), standard deviation (    2.87)
7
8   ArrayStatistics_externArray .....
9   Total ( 35) integer data :
10           1       2       3       4       5       6       7       8       9      10
11       11      12      13      14      15      16      17      18      19      20
12       21      22      23      24      25      26      27      28      29      30
13       31      32      33      34      35
14  min (  1), max ( 35), sum (  630.00), average (   18.00), variance (  102.00), standard deviation (   10.10)
15
16  ArrayStatistics_fileDataArray .....
17  Total ( 20) integer data :
18           1       2       3       4       5       6       7       8       9      10
19       11      12      13      14      15      16      17      18      19      20
20
21  min (  1), max ( 20), sum (  210.00), average (   10.50), variance (   33.25), standard deviation (    5.77)
22
23  ArrayStatistics_inputArray .....
24  Total ( 10) integer data :
25         10       9       8       7       6       5       4       3       2       1
26
27  min (  1), max ( 10), sum (   55.00), average (    5.50), variance (    8.25), standard deviation (    2.87)
28
29  ArrayStatistics_genBigRandArray .....
30  Total (1000000) integer data :
31      43772  992633  524109  333728  118867  451200  582276  428751   87061  962571
32   429215  972163  970055  199705  125982  744326  967430  434078  807127  314710
33
34     . . . .
35   638262  864460  689656  472410  918082  875151  482396  100257  780860  896350
36    19128  720252  294787  294308  923840  849253  925318  466734  520854  890764
37
38  min (  0), max (999999), sum (499999500000.00), average (499999.50), variance (83333333333.79), standard deviation (288675.13)
39
```

# Oral Test 5

# 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 - #endif를 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 중복되어 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.)