

프로그래밍언어

5. 전처리기, 다중 소스파일 프로그램 문자열, 파일 입출력



교수 김 영 탁

영남대학교 정보통신공학과

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

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

Outline

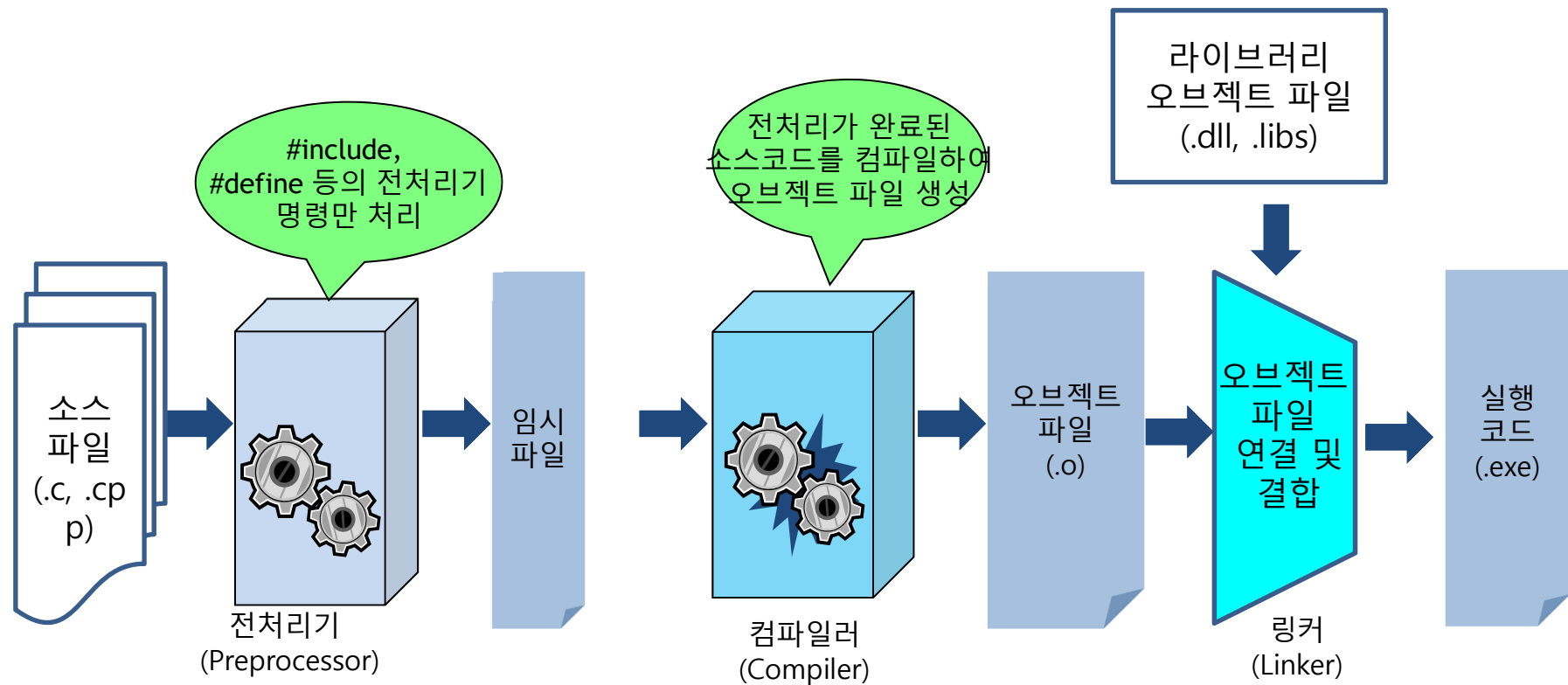
- ◆ 전 처리기 (Preprocessor)
- ◆ 기호상수 (Symbolic Constant)
- ◆ 매크로 (Macro)
- ◆ 다중 소스 파일
- ◆ 헤더 파일 (.h)
- ◆ 다중소스파일 구조의 프로그램 - handling big array
- ◆ 문자열 (string)
- ◆ 파일 입력 및 출력 (file input / output)



전처리기

전처리기란?

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



전처리기 (preprocessor) 지시어

전처리기 지시어 (예)	의미
#include <stdio.h> #include "MyHeaderFile.h"	지정된 파일 (헤더파일)을 포함
#define PI 3.141592 #define SQUARE(x) ((x) * (x)) #define MAX(x, y) ((x) >= (y)) ? (x) : (y))	기호 상수의 값을 지정 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, #endif가 나타날 때까지의 구간을 실행
#ifndef ARRAY_H	해당 기호상수가 지정되어 있지 않으면, #endif가 나타날 때까지의 구간을 실행
#line	행 번호를 출력
#elif	else-if를 의미
#pragma	시스템에 따라 다른 의미를 부여



전처리기와 기호상수를 사용한 프로그램 예

```
/* C Program with Preprocessor Definitions (1) */
#include <stdio.h>
#include <limits.h>
#define NUM_DATA 10
/*
#define INT_MAX // defined in <limits.h>
#define INT_MIN // defined in <limits.h>
*/
void main()
{
    int data;
    int array[NUM_DATA] = { 0 };
    int max = INT_MIN;
    int min = INT_MAX;
    int sum = 0;
    double average;
    int count = 0;
    printf("Input %d integer data: ", NUM_DATA);
```

```
/* C Program with Preprocessor Definitions (2) */

for (int i = 0; i < NUM_DATA; i++)
{
    scanf("%d", &data);
    array[i] = data;
    if (data > max)
        max = data;
    if (data < min)
        min = data;
    sum = sum + data;
    count++;
}
average = sum / (double)NUM_DATA;
printf("Input data: ");
for (int i = 0; i < count; i++)
    printf("%d ", array[i]);
printf("\nMax = %d, Min = %d\n",
        max, min);
printf("Sum = %d, Avg = %5.2lf\n",
        sum, average);
}
```



<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 0 if /j option used	(부호있는) 문자형의 최소값; /j 옵션이 사용되면 0
CHAR_MAX	127 255 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 (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 (0xFFFFFFFF)	부호없는 Long 형 정수의 최대값
_I64_MIN	-9223372036854775808	__int64형 (64비트) 정수의 최소값
_I64_MAX	9223372036854775807	__int64형 (64비트) 정수의 최대값
_UI64_MAX	18446744073709551615 (0xFFFFFFFFFFFFFFFF)	부호없는 __int64형 (64비트) 정수의 최대값



단순 매크로

- ◆ 단순 매크로(macro): 숫자 상수를 기호 상수 (symbolic constant)로 만든 것
- ◆ (예)

```
#define PI      3.141592      // 원주율
#define TWOPI   (3.141592 * 2.0) // 원주율의 2배
#define MAX_INT 2147483647    // 최대정수
#define EOF     (-1)         // 파일의 끝표시
#define MAX_STUDENTS 2000     // 최대 학생수
#define EPS     1.0e-9        // 실수의 계산 한계
#define DIGITS   "0123456789" // 문자 상수 정의
#define BRACKET  "{}[]"       // 문자 상수 정의
#define getchar() getc(stdin) // stdio.h에 정의
#define putchar() putc(stdout) // stdio.h에 정의
```



단순 매크로의 장점

- ◆ 프로그램의 가독성 (readability)을 높인다.
- ◆ 상수의 변경이 용이하다.

```
#include <stdio.h>

int main(void)
{
    . . . .
    krw1 = 1128 * usd1;
    . . . .
    krw2 = 1128 * usd2;
    . . . .
}
```

(a) 숫자 상수를 사용하는 경우
(숫자가 사용된 모든 곳을
수정하여야 함)

```
#include <stdio.h>
#define EXCHANGE_RATE 1128

int main(void)
{
    . . . .
    krw1 = EXCHANGE_RATE * usd1;
    . . . .
    krw2 = EXCHANGE_RATE * usd2;
    . . . .
}
```

(b) 기호 상수를 사용하는 경우
(기호상수가 정의된 곳만
수정하면 됨)



함수 매크로

◆ 함수 매크로(function-like macro) 란 매크로가 함수처럼 매개 변수를 가지는 것

```
#define SQUARE(x)    ((x) * (x))  
#define CUBIC(x)     ((x) * (x) * (x))  
#define SUM(x, y)    ((x) + (y))  
#define AVERAGE(x, y, z) (( (x) + (y) + (z) ) / 3 )  
#define MAX(x,y)     ( (x) > (y) ) ? (x) : (y)  
#define MIN(x,y)     ( (x) < (y) ) ? (x) : (y)
```

◆ 주의할 점

```
#define SQUARE(x)    x*x           // 위험 !!  
v = SQUARE(a+b);  
→ v = a + b*a + b;
```



```
#define SQUARE(x)    ((x)*(x))     // 올바른 형태
```

함수 매크로의 장단점

◆ 함수 매크로의 장단점

- 함수 호출 단계가 필요 없어 실행 속도가 빠르다.
- 소스 코드의 길이가 길어진다.

◆ 간단한 기능은 매크로를 사용

```
#define MIN(x, y)    ((x) < (y) ? (x) : (y))  
#define ABS(x)      ((x) > 0 ? (x) : -(x))
```

◆ 매크로를 한 줄 이상 연장하는 방법

- #define PRINT(x) if(debug==1 && mode==1) \
printf("%d", x);



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 번호 로 치환된다.
__FILE__	이 매크로를 만나면 소스 파일 이름 으로 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 설정

◆ GET_BIT(w, k)는 변수 w에서 k번째 비트의 값을 0 또는 1로 반환

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

◆ SET_BIT_ON(w, k)는 변수 w의 k번째 비트를 1로 설정하는 매크로

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

◆ SET_BIT_OFF(w, k)는 변수 w의 k번째 비트를 0로 설정하는 매크로

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



Bit 관련 매크로 사용 프로그램 예

```
#include <stdio.h>

#define GET_BIT(w, k) (((w) >> (k)) & 0x01)
#define SET_BIT_ON(w, k) ((w) |= (0x01 << (k)))
#define SET_BIT_OFF(w, k) ((w) &= ~(0x01 << (k)))

int main(void)
{
    int data=0;
    SET_BIT_ON(data, 2);
    printf("%08X\n", data);
    printf("%d\n", GET_BIT(data, 2));

    SET_BIT_OFF(data, 2);
    printf("%08X\n", data);
    printf("%d\n", GET_BIT(data, 2));
    return 0;
}
```



00000004
1
00000000
0



#ifdef

◆ 어떤 조건이 만족되었을 경우에만 컴파일 하는 조건부 컴파일 지시

```
#ifdef MACRO_A
문장1    // MACRO_A가 정의되었을 경우
...
#else
문장2    // MACRO_A가 정의되지 않았을
경우
...
#endif
```

#define DEBUG_1

컴파일에서
포함됨

```
double average(int x, int y)
{
#ifdef DEBUG_1
    printf("x=%d, y=%d\n", x, y);
#endif
    return (x + y) / 2.0;
}
```

// #define DEBUG_1

DEBUG_1이
주석처리되어
컴파일에서
포함되지 않음

```
double average(int x, int y)
{
#ifdef DEBUG_1
    printf("x=%d, y=%d\n", x, y);
#endif
    return (x + y) / 2.0;
}
```



#ifdef을 사용한 조건부 컴파일의 예

```
/* main.c */
#include <stdio.h>
#include "Parameters.h"
#include "Array.h"
#include "File_IO_Array.h"
// #define DEBUG_with_Console_Output
void main()
{
    int int_array[MAX_NUM_DATA] = { 0 };
    int num_data = 0, sum = 0;
    const char *input_file_name = "Input_data.txt";
    const char *output_file_name = "Output_data.txt";
    FILE *fin, *fout;
    fin = file_open(input_file_name, "r");
    fout = file_open(output_file_name, "w");
    num_data = fileInputArray(fin, int_array, MAX_NUM_DATA);
    #ifdef DEBUG_with_Console_Output
        printf("Input data : \n");
        printArray(int_array, num_data, DATA_PER_LINE);
    #endif
    sum = sumArray(int_array, num_data);
    #ifdef DEBUG_with_Console_Output
        printf("Sum of %d input data (from file %s) is %d\n", num_data, input_file_name, sum);
    #endif
    fileOutputArray(fout, int_array, num_data, DATA_PER_LINE);
    fclose(fin);
    fclose(fout);
}
```



#ifndef, #undef

◆ #ifndef

- 어떤 매크로가 정의되어 있지 않으면 컴파일에 포함된다.

```
#ifndef LIMIT
#define LIMIT 1000
#endif
```

LIMIT가 정의되어 있지 않으면

LIMIT를 정의해준다.

◆ #undef

- 매크로의 정의를 취소한다

```
#define SIZE 100
...
#undef SIZE
#define SIZE 200
```

SIZE의 정의를 취소한다.



#if

- ◆ 기호가 참으로 계산되면 컴파일
- ◆ 조건은 상수이어야 하고 논리, 관계 연산자 사용 가능
- ◆ 형식

```
#if Condition
    statement_1;
    statement_2;
#endif
```

- ◆ 설명

- 만약 condition이 참이면 statement 들이 컴파일 되고, 프로그램 실행에 포함된다.

- ◆ 예)

```
#if DEBUG == 1
    printf("Current value is %d\n", value);
#endif
```



#if-#else-#endif

◆ 형식

```
#if Condition_1
    statement_1
#elif Condition_2
    statement_2
#else
    statement_3
#endif
```

◆ (예)

```
#if NATION == 1
#include "korea.h"
#elif NATION == 2
#include "china.h"
#else
#include "usa.h"
#endif
```



#if를 사용한 조건부 컴파일

```
#if (VERSION > 3)    // 가능! 버전이 3 이상이면 컴파일
...
#endif

#if (AUTHOR == KIM)  // 가능!! KIM은 다른 매크로
#if (VERSION*10 > 500 && LEVEL == BASIC)    // 가능!!

#if (VERSION > 3.0)    // 오류 !! 버전 번호는 300과 같은 정수로
표시
#if (AUTHOR == "CHULSOO") // 오류 !!

#if (VERSION > 300 || defined(DELUXE) )
```



#ifdef으로 지정되는 조건부 컴파일을 이용하는 디버깅

```
#define DEBUG 1
```

```
...
```

```
#if DEBUG == 1
```

```
    printf("현재 counter의 값은 %d입니다.\n", counter);
```

```
#endif
```

```
#define DEBUG
```

```
...
```

```
#ifdef DEBUG
```

```
printf("현재 counter의 값은 %d입니다.\n", counter);
```

```
#endif
```

```
...
```

```
#if defined(DEBUG)
```

```
printf("현재 counter의 값은 %d입니다.\n", counter);
```

```
#endif
```



#if, #elif를 사용한 알고리즘의 선택 예

◆ 정렬 알고리즘을 선택

```
#define SORT_METHOD 3

#if (SORT_METHOD == 1)
...    // 선택정렬구현
#elif (SORT_METHOD == 2)
...    // 버블정렬구현
#else
...    // 퀵정렬구현
#endif
```



다중 소스 파일

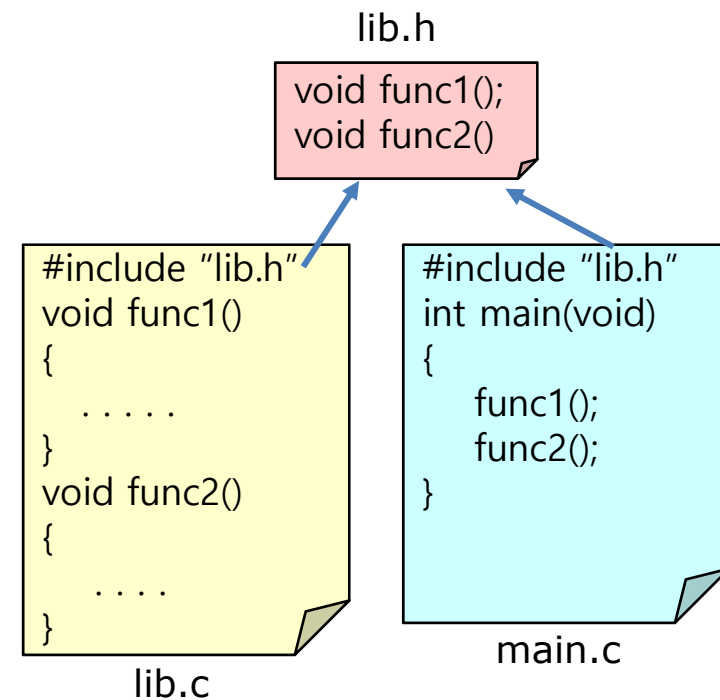
다중 소스 파일

◆ 단일 소스 파일

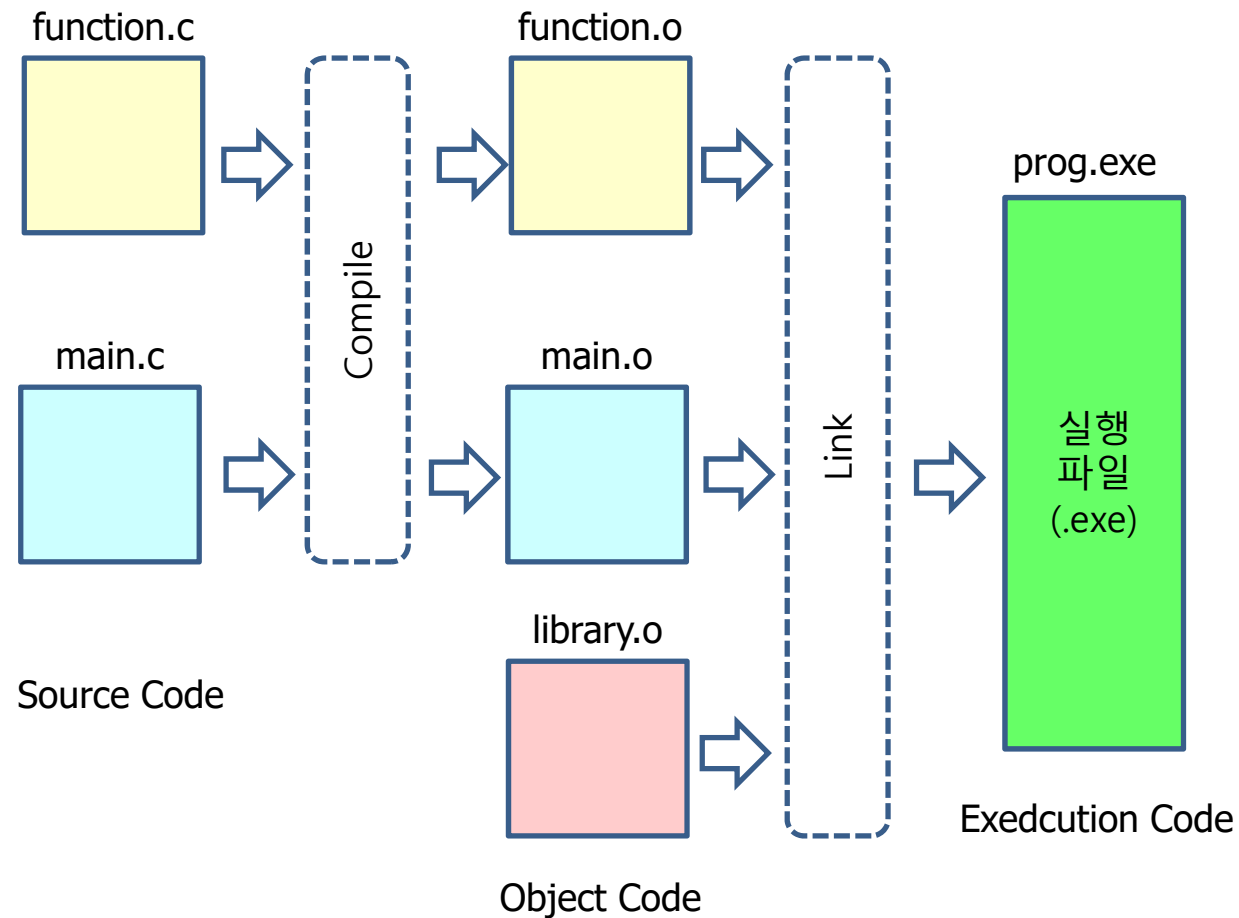
- 파일의 크기가 너무 커진다.
- 소스 파일을 다시 사용하기가 어려움

◆ 다중 소스 파일

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



다중 소스 파일의 컴파일과 링크



예제

main.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

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

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

power.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

```
void draw_line(...)  
{  
    ....  
}  
void draw_rect(...)  
{  
    ....  
}  
void draw_circle(...)  
{  
    ....  
}
```

graphics.c

공급자

함수 원형 정의가 중복되어 있음

```
void draw_line(...);  
void draw_rect(...);  
void draw_circle(...);
```

```
int main(void)  
{  
    draw_rect(...);  
    draw_circle(...);  
    ...  
    return 0;  
}
```

main.c

사용자

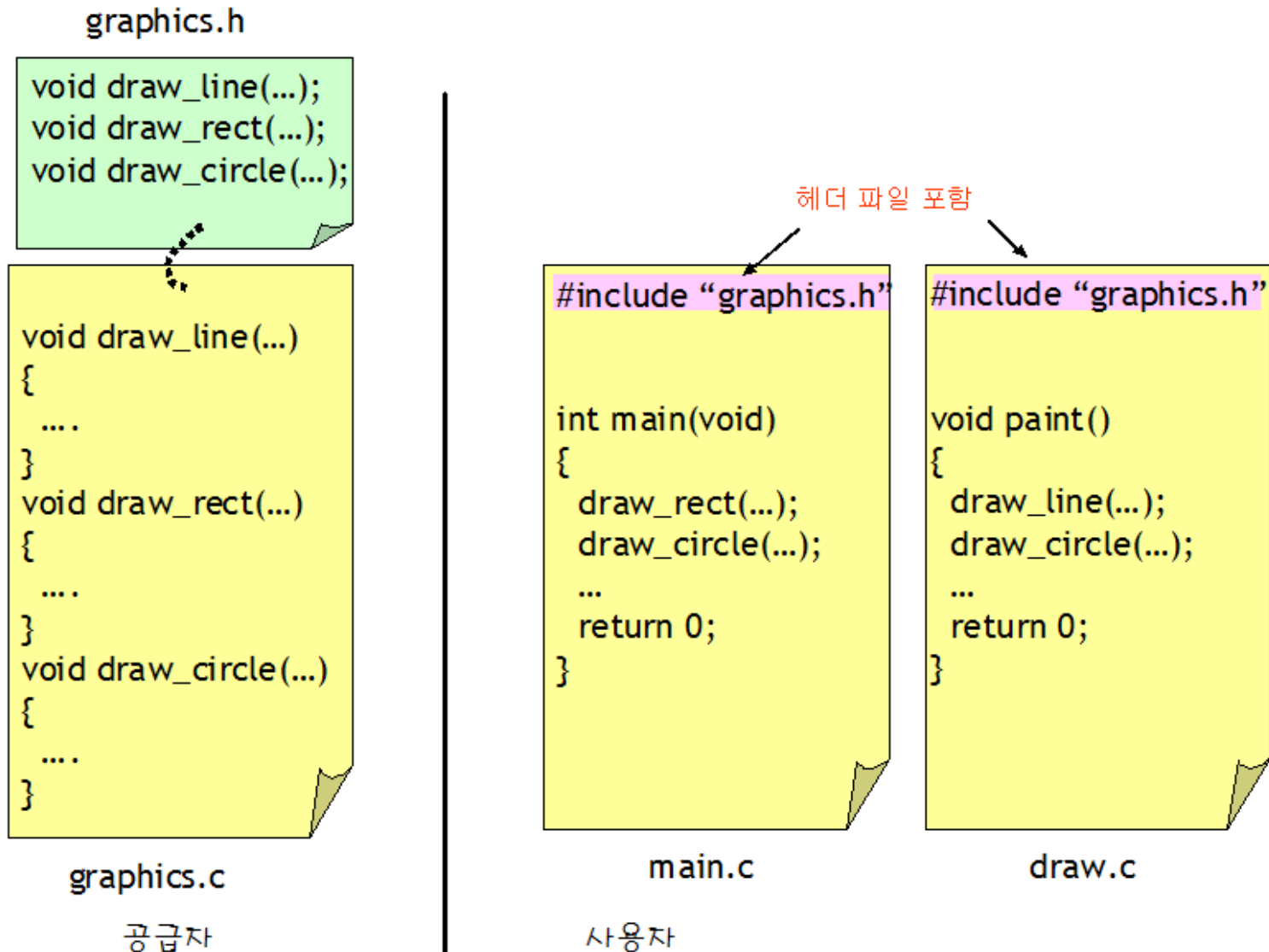
```
void draw_line(...);  
void draw_rect(...);  
void draw_circle(...);
```

```
void paint()  
{  
    draw_line(...);  
    draw_circle(...);  
    ...  
    return 0;  
}
```

draw.c



헤더 파일을 사용하면



다중 소스 파일에서 외부에서 선언된 전역 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) 선언 전역 변수의 사용



헤더 파일 이중 포함 방지

```
#include <stdio.h>
```

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

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

구조체의 정의가 이중으로 포함
되어서 오류가 발생한다.

```
#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
```

```
}
```



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.



다중 소스 파일 프로그램의 예
- **big array** 생성 및 데이터 분석, 처리

배열관련 헤더파일, 데이터 파일

◆ **BigArray.h**

```
/* BigArray.h*/

#ifndef BIG_ARRAY_H
#define BIG_ARRAY_H

#include <stdio.h>

void printBigArray(int *array, int size, int line_size);
void fprintfBigArray(FILE *fout, int *array, int size, int line_size);
void printBigArraySample(int* array, int size, int items_per_line, int num_sample_lines);
void fprintfBigArraySample(FILE* fout, int *array, int size, int items_per_line,
    int num_sample_lines);
double sumArray(int *array, int size);
void genBigRandArray(int* array, int bigRandSize, int base);
void suffleBigArray(int *array, int size);
void selectionSort(int *array, int size);
void getArrayStatistics(int *array, int num_data);
void fgetArrayStatistics(FILE *fout, int *array, int num_data);
#endif
```



BigArray.c

```
/* BigArray.c (1) */
```

```
#include <stdio.h>
#include <stdlib.h> // for calloc()
#include <time.h>   // for time(NULL)
#include <math.h>   // for sqrt()
#include "BigArray.h"
```

```
void printArray(int *array, int size, int line_size)
```

```
{
    for (int i = 0; i < size; i++)
    {
        printf("%5d ", array[i]);
        if ((i+1) % line_size == 0)
            printf("\n");
    }
    printf("\n");
}
```

```
void fprintfArray(FILE *fout, int *array, int size, int line_size)
```

```
{
    for (int i = 0; i < size; i++)
    {
        fprintf(fout, "%5d", array[i]);
        if ( (i+1) % line_size == 0)
            fprintf(fout, "\n");
    }
    fprintf(fout, "\n");
}
```



```
void printBigArraySample(int *array, int size, int items_per_line = 10, int num_sample_lines = 3)
```

```
{
    int count = 0;
    for (int i = 0; i < num_sample_lines; i++)
    {
        for (int j = 0; j < items_per_line; j++)
        {
            if (count > size)
            {
                printf("\n");
                return;
            }
            printf("%8d ", array[count]);
            count++;
        }
        printf("\n");
    }

    if (count < (size - items_per_line * num_sample_lines))
        count = size - items_per_line * num_sample_lines;
    if (count >= size)
        return;
    printf("\n      . . . . . \n");

    for (int i = 0; i < num_sample_lines; i++)
    {
        for (int j = 0; j < items_per_line; j++)
        {
            if (count > size)
            {
                printf("\n");
                return;
            }
            printf("%8d ", array[count]);
            count++;
        }
        printf("\n");
    }
    printf("\n");
}
```

Test Array Handling (1: data_array; 2: extern_array; 3: data_file; 4: data_input; 5: genBigArray; Esc: terminate) : 5

Input size of big array (more than 100000) = 5000000

1933149	4629579	677214	4995449	4658044	1748393	1300266	957945	3408938	1050798
4354568	4833351	3966216	1431627	2225885	677605	642459	4279665	3398895	4851543
2252622	1514904	1922667	2579686	74733	111865	3833824	274079	2932203	3678739

2880895	2613302	4443570	477458	2860169	56872	2486445	1953775	843418	2226505
1965962	765157	2634118	70258	3169227	3856203	3446590	1611687	3384393	736292
3037865	2652578	2784692	598118	4479926	1169007	2841378	3398545	3740914	269878

Test Array Handling (1: data_array; 2: extern_array; 3: data_file; 4: data_input; 5: genBigArray; Esc: terminate) : 5

Input size of big array (more than 100000) = 10000000

8216638	6014765	6551043	6233595	1114979	5000197	8490768	6413263	6342849	6483787
2708122	6975471	785256	3058570	3707794	3735000	5660365	7747284	671630	2703437
2742725	897084	6377672	3854037	7970317	9687132	3175325	7335482	8890053	9790544

3828663	5358052	1208364	6213058	7829338	9704686	1975527	5214530	9814114	9815940
291419	9232825	4231297	5359029	3919781	5621412	6314177	6722889	1595807	8080207
4890558	9056460	348732	8827801	3979937	4459108	3439078	6507950	9660908	9852874



```
/* BigArray.c (3) */
```

```
double sumArray(int *array, int size)
```

```
{  
    double sum = 0.0; // local variable  
    for (int i = 0; i < size; i++)  
        sum += array[i];  
    return sum; // return the result  
}
```

```
void genBigRandArray(int *array, int bigRandSize, int base)
```

```
{  
    char* flag;  
    int count = 0;  
    unsigned int u_int32 = 0;  
    unsigned int bigRand, bigRand_withOffset;  
  
    flag = (char*)calloc(bigRandSize, sizeof(char));  
    while (count < bigRandSize)  
    {  
        u_int32 = ((long)rand() << 15) | rand();  
        bigRand = u_int32 % bigRandSize;  
        if (flag[bigRand] == 1) {  
            continue;  
        }  
        else {  
            flag[bigRand] = 1;  
            bigRand_withOffset = bigRand + base;  
            array[count++] = bigRand_withOffset;  
        }  
    }  
    free(flag);  
}
```



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

```
void getArrayStatistics(int *array, int size)
```

```
{
```

```
    int data, min, max;
```

```
    double sum = 0.0, var, diff, sq_diff_sum = 0.0, avg, std_dev;
```

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

```
    {
```

```
        data = array[i];
```

```
        sum += data;
```

```
        if (i == 0) {
```

```
            min = max = array[0];
```

```
        } else {
```

```
            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);
```

```
    printf("Total (%3d) integer data : \n", size);
```

```
    printArray(array, size, 10);
```

```
    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);
```

```
}
```



```
/* Array.c (5) */
```

```
void suffleBigArray(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 array*/  
        d = array[i1];  
        array[i1] = array[i2];  
        array[i2] = d;  
    }  
}
```



ArrayData

◆ ArrayData.c

```
/* ArrayData.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
};
```

◆ input_data.txt

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



다중 소스코드 프로그램 예제 – main_BigArray.c

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

#define ESC 0x1B
#define MAX_NUM_DATA 100

void arrayStatistics_basicArray(FILE* fout);
void arrayStatistics_externArray(FILE* fout);
void arrayStatistics_fileDataArray(FILE* fout);
void arrayStatistics_inputArray(FILE* fout);
void test_genBigArray();

int main(int argc, char argv[])
{
    int num_data, data;
    FILE* fout;
    char menu;

    fout = fopen("array_output.txt", "w");
    if (fout == NULL)
    {
        printf("Error in creation of array_output.txt !!\n");
        return -1;
    }
}
```




```

/* main_BigArray.cpp (2) */

while (1)
{
    printf("\nTest Array Handling (1: data_array; 2: extern_array;
        3: data_file; 4: data_input; 5: genBigArray; 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':
            test_genBigArray();
            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[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.\n");
}

```

```

Test Array Handling (1: data_array; 2: extern_array; 3: data_file; 4: data_input; 5: genBigArray; 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.

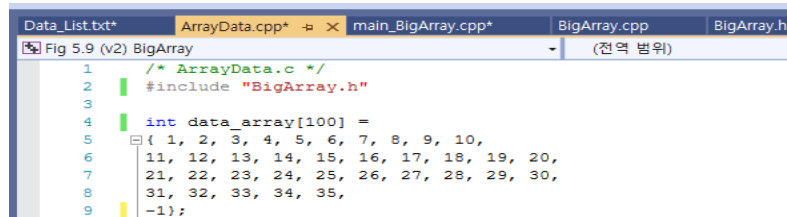
```



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

```
void arrayStatistics_externArray(FILE* fout)
```

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



```
1  /* ArrayData.c */  
2  #include "BigArray.h"  
3  
4  int data_array[100] =  
5  { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  
6   11, 12, 13, 14, 15, 16, 17, 18, 19, 20,  
7   21, 22, 23, 24, 25, 26, 27, 28, 29, 30,  
8   31, 32, 33, 34, 35,  
9   -1};
```

Test Array Handling (1: data_array; 2: extern_array; 3: data_file; 4: data_input; 5: genBigArray; 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.



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

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

```
Test Array Handling (1: data_array; 2: extern_array; 3: data_file; 4: data_input; 5: genBigArray; Esc: terminate) : 4
ArrayStatistics_inputArray .....
Please input the number of integers (less than 100) = 20
Input 20 integer data : 9 8 7 6 5 4 3 2 1 0 11 12 13 14 15 19 18 17 16 10
Total ( 20) integer data :
    9    8    7    6    5    4    3    2    1    0
   11   12   13   14   15   19   18   17   16   10

min ( 0), max ( 19), sum ( 190.00), average ( 9.50), variance ( 33.25), standard deviation ( 5.77)
arrayStatistics_inputArray - completed. Result is also stored in output file.
```



void test_genBigArray()

```
{
    int* bigArray;
    int bigArray_size;

    printf("\nInput size of big array (more than 100000) = ");
    scanf("%d", &bigArray_size);
    bigArray = (int*)calloc(bigArray_size, sizeof(int));
    genBigRandArray(bigArray, bigArray_size, 0);
    printBigArraySample(bigArray, bigArray_size, 10, 3);
}
```

Test Array Handling (1: data_array; 2: extern_array; 3: data_file; 4: data_input; 5: genBigArray; Esc: terminate) : 5

Input size of big array (more than 100000) = 5000000

1933149	4629579	677214	4995449	4658044	1748393	1300266	957945	3408938	1050798
4354568	4833351	3966216	1431627	2225885	677605	642459	4279665	3398895	4851543
2252622	1514904	1922667	2579686	74733	111865	3833824	274079	2932203	3678739

2880895	2613302	4443570	477458	2860169	56872	2486445	1953775	843418	2226505
1965962	765157	2634118	70258	3169227	3856203	3446590	1611687	3384393	736292
3037865	2652578	2784692	598118	4479926	1169007	2841378	3398545	3740914	269878

Test Array Handling (1: data_array; 2: extern_array; 3: data_file; 4: data_input; 5: genBigArray; Esc: terminate) : 5

Input size of big array (more than 100000) = 10000000

8216638	6014765	6551043	6233595	1114979	5000197	8490768	6413263	6342849	6483787
2708122	6975471	785256	3058570	3707794	3735000	5660365	7747284	671630	2703437
2742725	897084	6377672	3854037	7970317	9687132	3175325	7335482	8890053	9790544

3828663	5358052	1208364	6213058	7829338	9704686	1975527	5214530	9814114	9815940
291419	9232825	4231297	5359029	3919781	5621412	6314177	6722889	1595807	8080207
4890558	9056460	348732	8827801	3979937	4459108	3439078	6507950	9660308	9852874



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 / Close의 예

```
/* File open, input and output */
#include <stdio.h>
. . . .

void main()
{
    . . . . .
    FILE *fout;

    fout = fopen("output.txt", "w");
    if (fout == NULL)
    {
        printf("Error in creation of output.txt file !!\n");
        exit(-1);
    }
    . . . . .
    for (int i = 0; i < size; i++)
    {
        fprintf(fout, "%5d ", array[i]);
        if ((i+1) % line_size == 0)
            fprintf(fout, "\n");
    }
    fprintf(fout, "\n");
    fclose(fout);
}
```



파일 읽기와 쓰기 – fprintf(), fscanf()

◆ 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, ...);
```

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

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



파일 입력, 출력 예제 프로그램 – 데이터 배열

```
int fgetArrayFromFile(const char* fin_name, int *array, int max_size)
{
    FILE* fin;
    int data, num_data = 0;
    fin = fopen(fin_name, "r");
    if (fin == NULL)
    {
        printf("Error in opening input data file !!\n");
        return 0;
    }
    while (fscanf(fin, "%d", &data) != EOF)
    {
        if ((data == -1) || (num_data >= max_size))
            break;
        array[num_data] = data;
        num_data++;
    }
    fclose(fin);
    return num_data;
}
```



```
void fprintBigArraySample(FILE* fout, int* array, int size, int items_per_line = 10,  
    int num_sample_lines = 3)
```

```
{
    int count = 0;
    for (int i = 0; i < num_sample_lines; i++)
    {
        for (int j = 0; j < items_per_line; j++)
        {
            if (count > size) {
                fprintf(fout, "\n");
                return;
            }
            fprintf(fout, "%8d ", array[count]);
            count++;
        }
        fprintf(fout, "\n");
    }

    if (count < (size - items_per_line * num_sample_lines))
        count = size - items_per_line * num_sample_lines;
    if (count >= size)
        return;

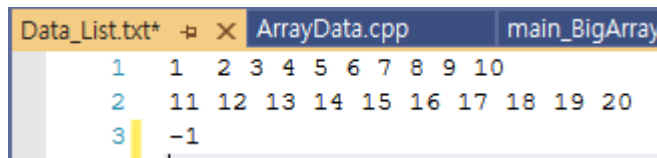
    fprintf(fout, " . . . . . \n");
    for (int i = 0; i < num_sample_lines; i++)
    {
        for (int j = 0; j < items_per_line; j++)
        {
            if (count > size) {
                fprintf(fout, "\n");
                return;
            }
            fprintf(fout, "%8d ", array[count]);
            count++;
        }
        fprintf(fout, "\n");
    }
    fprintf(fout, "\n");
}
```

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");

    num_data = fgetArrayFromFile("Data_List.txt", data_array, MAX_NUM_DATA);
    getArrayStatistics(data_array, num_data);
    fgetArrayStatistics(fout, data_array, num_data);
    printf("arrayStatistics_fileDataArray - completed. Result is also stored in output file.\n");
}
```



The screenshot shows a code editor with three tabs: Data_List.txt*, ArrayData.cpp, and main_BigArray. The Data_List.txt tab is active, displaying the following content:

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

```
Test Array Handling (1: data_array; 2: extern_array; 3: data_file; 4: data_input; 5: genBigArray; 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

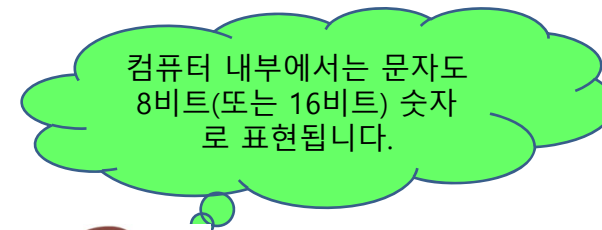
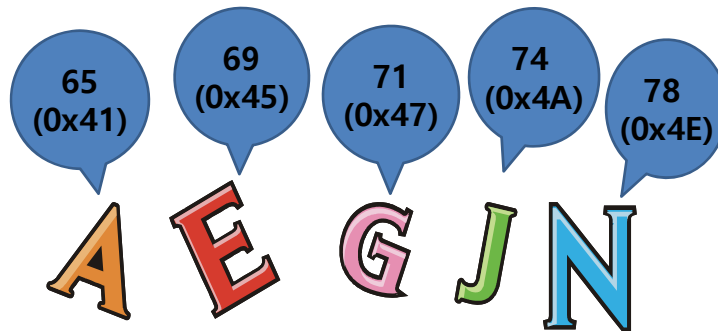
min ( 1), max ( 20), sum ( 210.00), average ( 10.50), variance ( 33.25), standard deviation ( 5.77)
arrayStatistics_fileDataArray - completed. Result is also stored in output file.
```



문자열 (string)

문자 (character) 표현방법

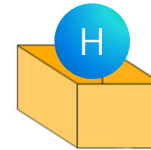
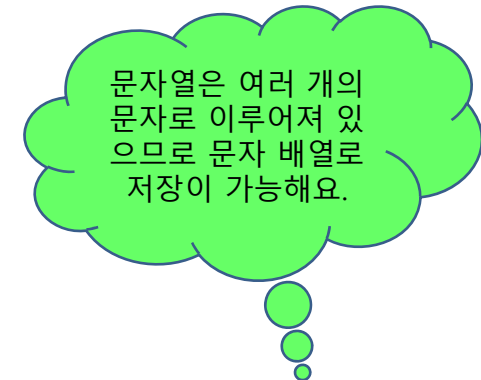
- ◆ 컴퓨터에서는 각각의 문자에 숫자코드를 붙여서 표시한다.
- ◆ **ASCII (American Standard Code for Information Interchange): 표준적인 8비트 문자코드**
 - 0에서 127까지의 숫자를 이용하여 문자 (영문자, 숫자, 특수문자 등)를 표현
- ◆ **유니코드(unicode): 표준적인 16비트 문자코드**
 - 전세계의 모든 문자를 일관되게 표현하고 다룰 수 있도록 설계



문자열 표현 방법

◆ 문자열(string): 문자 (character)들이 여러 개 모인 것

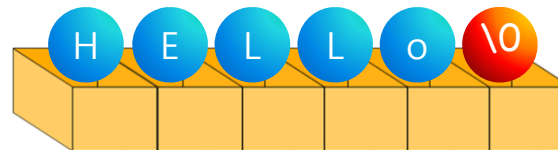
- "A"
- "Hello World!"
- "변수 score의 값은 %d입니다"



하나의 문자는 char형 변수로 저장

◆ 문자열의 저장

- 문자 배열 (char array)사용



문자열은 char형 배열로 저장

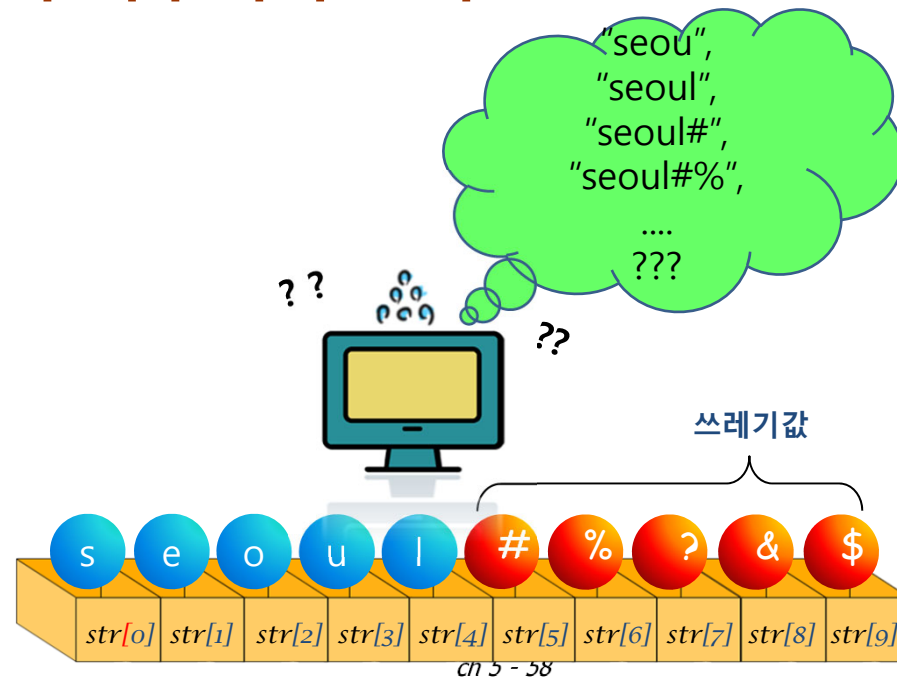


NULL 문자

◆ NULL 문자: 문자열의 끝을 나타낸다.



◆ 문자열은 어디서 종료되는지 알 수 없으므로 NULL 문자로 표시 해주어야 한다.



문자 변수와 문자 상수

```
// 문자 상수
#include <stdio.h>

int main(void)
{
    char code1 = 'A';
    char code2 = 65;

    printf("code1=%c, code1=%d\n", code1, code1);
    printf("code2=%c, code2=%d\n", code2, code2);
    return 0;
}
```

문자변수

문자상수

A

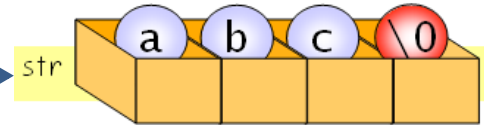
65
(0x41)

code1=A, code1=65
code2=A, code2=65

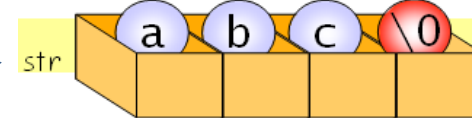


문자 배열의 초기화

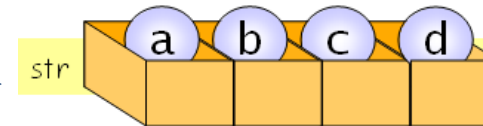
◆ `char str[4] = { 'a', 'b', 'c', '\0' };`



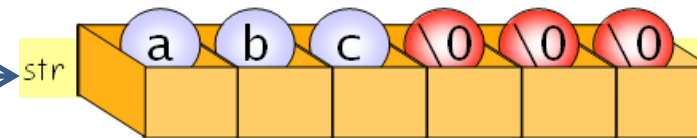
◆ `char str[4] = "abc";`



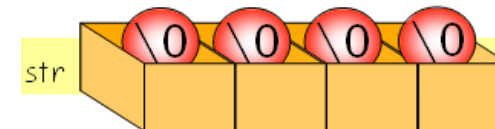
◆ `char str[4] = "abcdef";`



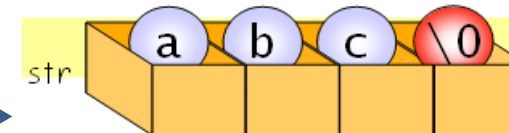
◆ `char str[6] = "abc";`



◆ `char str[4] = "";`



◆ `char str[] = "abc";`



문자 배열에 문자열을 저장

1. 각각의 문자 배열 원소에 원하는 문자를 개별적으로 대입하는 방법이다.

- `str[0] = 'W';`
- `str[1] = 'o';`
- `str[2] = 'r';`
- `str[3] = 'l';`
- `str[4] = 'd';`
- `str[5] = '\0';`

2. `strcpy()`를 사용하여 문자열을 문자 배열에 복사

- `strcpy(str, "World");`



문자열 상수

- ◆ 문자열 상수: "HelloWorld"와 같이 프로그램 소스 안에 포함된 문자열
- ◆ 문자열 상수는 메모리 영역 중에서 텍스트 세그먼트(text segment)에 저장

```
const char *p = "HelloWorld";
```

위 문장의
정확한 의미
는 무엇일까
요?



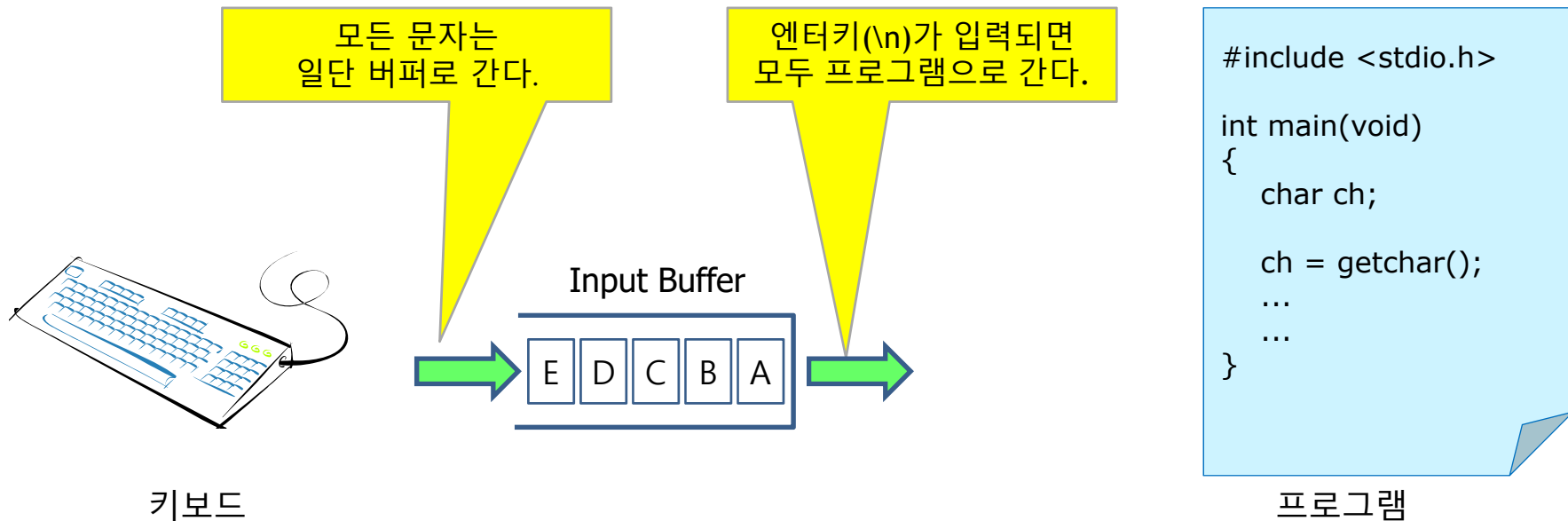
문자 입출력 라이브러리

입출력 함수	필요한 헤더 파일	설명
<code>int getchar(void)</code>	<code><stdio.h></code>	하나의 문자를 읽어서 반환한다.
<code>void putchar(int c)</code>	<code><stdio.h></code>	변수 <code>c</code> 에 저장된 문자를 출력한다.
<code>int getch(void)</code> <code>int _getch(void);</code>	<code><conio.h></code>	하나의 문자를 읽어서 반환한다 (버퍼를 사용하지 않음).
<code>void putch(int c)</code>	<code><conio.h></code>	변수 <code>c</code> 에 저장된 문자를 출력한다 (버퍼를 사용하지 않음).
<code>scanf("%c", &c)</code>	<code><stdio.h></code>	하나의 문자를 읽어서 변수 <code>c</code> 에 저장한다.
<code>printf("%c", c);</code>	<code><stdio.h></code>	변수 <code>c</code> 에 저장된 문자를 출력한다.



버퍼링을 사용한 문자입력

◆ 엔터키 (enter key)를 쳐야만 입력을 받는 이유



getch(), getche(), getchar()

	헤더파일	입력 버퍼 사용여부	에코 여부	응답성	문자수정 여부
getchar()	<stdio.h>	사용함 (엔터키를 눌러야 입력됨)	에코 (echo)	줄단위	가능
getch() _getch()	<conio.h>	사용하지 않음	에코 하 지 않음	문자단위	불가능
getche()	<conio.h>	사용하지 않음	에코	문자단위	불가능



용도에 맞는 것을 골라
사용하세요!

버퍼가 없이 **바로 받으려면**
getch()를 사용합니다.



문자열 입출력 라이브러리 함수

입출력 함수	설명
<code>int scanf("%s", s)</code>	문자열을 읽어서 문자배열 <code>s[]</code> 에 저장
<code>int printf("%s", s)</code>	배열 <code>s[]</code> 에 저장되어 있는 문자열을 출력한다.
<code>char *gets(char *s)</code>	한 줄의 문자열을 읽어서 문자 배열 <code>s[]</code> 에 저장한다.
<code>int puts(const char *s)</code>	배열 <code>s[]</code> 에 저장되어 있는 한 줄의 문자열을 출력한다.



문자 처리 라이브러리 함수

◆ 문자를 검사하거나 문자를 변환한다.

함수	설명
isalpha(c)	c가 영문자인가?(a-z, A-Z)
isupper(c)	c가 대문자인가?(A-Z)
islower(c)	c가 소문자인가?(a-z)
isdigit(c)	c가 숫자인가?(0-9)
isalnum(c)	c가 영문자이나 숫자인가?(a-z, A-Z, 0-9)
isxdigit(c)	c가 16진수의 숫자인가?(0-9, A-F, a-f)
isspace(c)	c가 공백문자인가?(' ', '\n', '\t', '\v', '\r')
ispunct(c)	c가 구두점 문자인가?
isprint(c)	C가 출력가능한 문자인가?
iscntrl(c)	c가 제어 문자인가?
isascii(c)	c가 아스키 코드인가?



문자 처리 라이브러리 함수

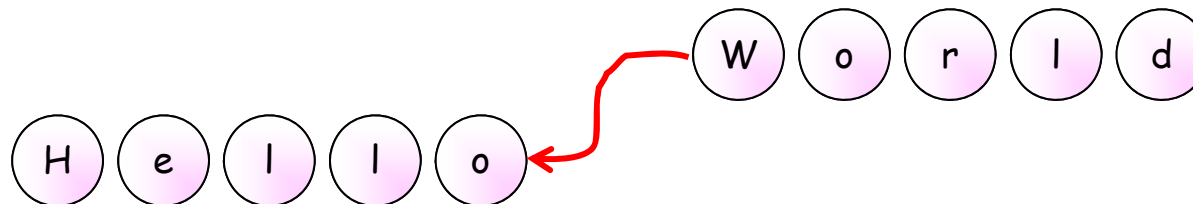
◆ 문자를 검사하거나 문자를 변환한다.

함수	설명
toupper(c)	c를 대문자로 바꾼다.
tolower(c)	c를 소문자로 바꾼다.
toascii(c)	c를 아스키 코드로 바꾼다.



문자열 처리 라이브러리

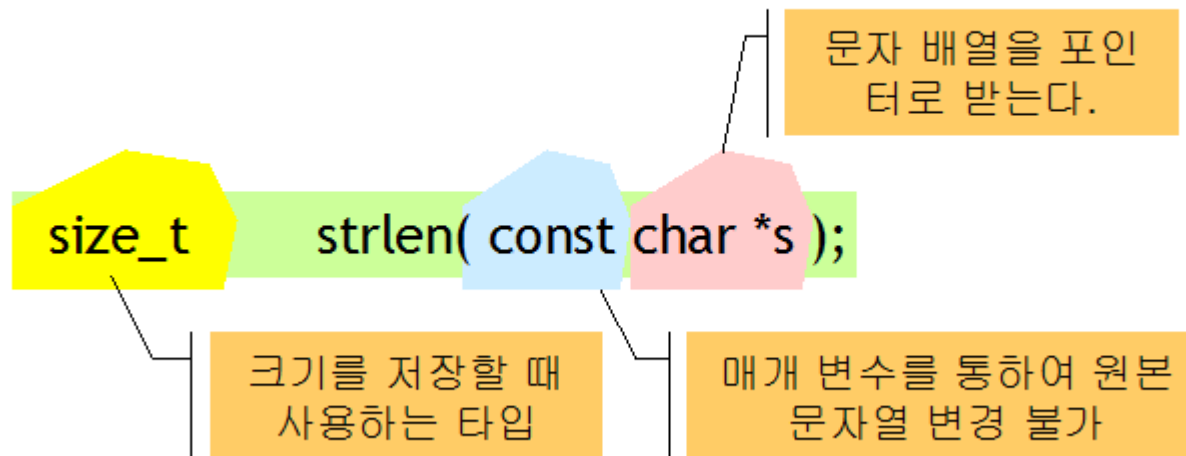
함수	설명
strlen(s)	문자열 s의 길이를 구한다.
strcpy(s1, s2)	s2를 s1에 복사한다.
strcat(s1, s2)	s2를 s1의 끝에 붙여넣는다.
strcmp(s1, s2)	s1과 s2를 비교한다.
strncpy(s1, s2, n)	s2의 최대 n개의 문자를 s1에 복사한다.
strncat(s1, s2, n)	s2의 최대 n개의 문자를 s1의 끝에 붙여넣는다.
strncmp(s1, s2, n)	최대 n개의 문자까지 s1과 s2를 비교한다.
strchr(s, c)	문자열 s안에서 문자 c를 찾는다.
strstr(s1, s2)	문자열 s1에서 문자열 s2를 찾는다.



문자열 길이 (strlen)

◆ 문자열의 길이

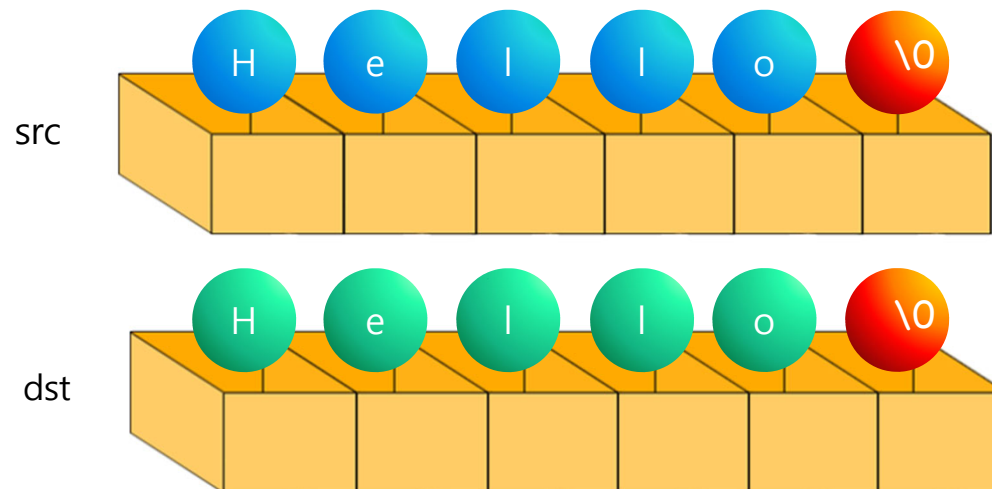
- strlen("Hello")는 5를 반환



문자열 복사 (strcpy)

◆ 문자열 복사

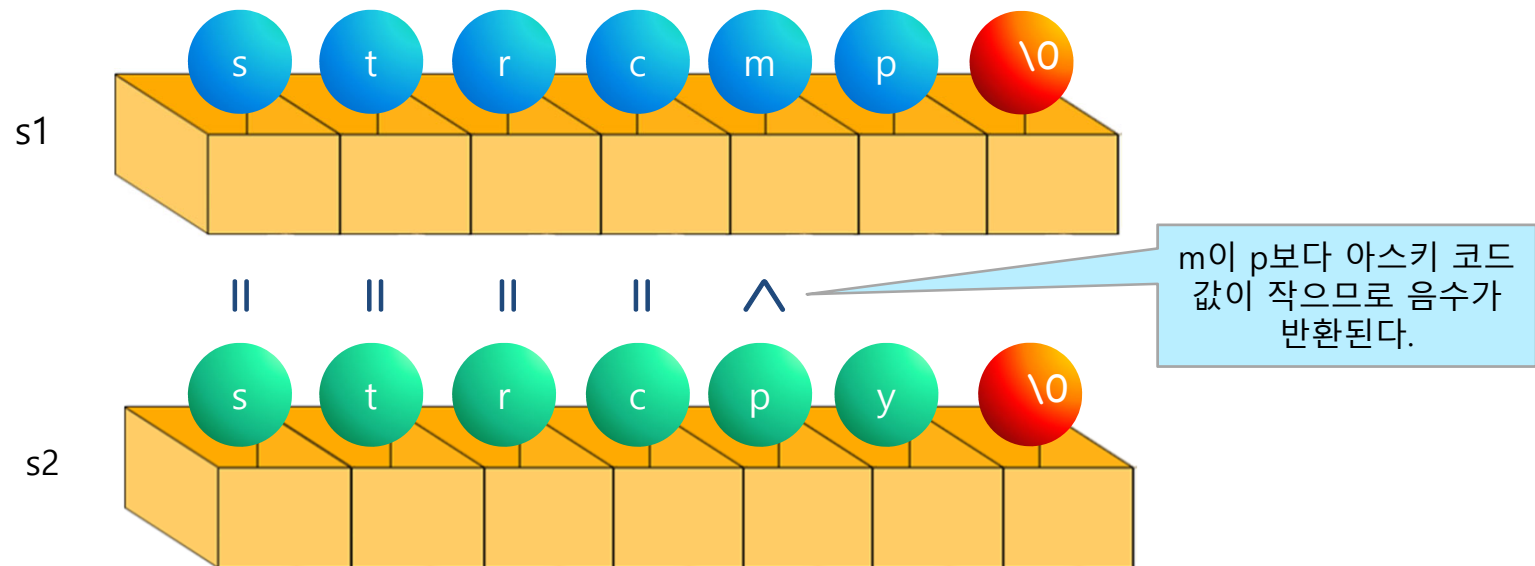
```
char dst[6];  
char src[6] = "Hello";  
strcpy(dst, src);
```



문자열 비교 (string compare)

```
int strcmp( const char *s1, const char *s2 );
```

반환값	s1과 s2의 관계
< 0	s1이 s2보다 작다
0	s1이 s2와 같다.
> 0	s1이 s2보다 크다.



파일 입력, 출력 예제 프로그램 – 문자열 배열

```
/** SimpleFileInputOutput.cpp */

#include <stdio.h>
#include <string.h>

#define MAX_WORD_LEN 50
#define NUM_WORDS 100
void fprintfStringArray(FILE *fout, char wordList[][MAX_WORD_LEN], int num_words, int words_per_line);

int main()
{
    FILE *fin = NULL;
    FILE *fout = NULL;

    char str[80];
    char wordList[NUM_WORDS][MAX_WORD_LEN];
    int word_count;

    fin = fopen("input.txt", "r");
    if (fin == NULL)
    {
        printf("Error in input data file open !!\n");
        return 0;
    }
}
```



```

/** FileInputOutput.cpp (2) */

fout = fopen("output.txt", "w");
if (fout == NULL)
{
    printf("Error in output data file creation !!\n");
    return 0;
}

word_count = 0;
while (fscanf(fin, "%s", str) != EOF)
{
    printf("%2d-th input word: %s\n", word_count, str);
    strcpy(wordList[word_count], str);
    word_count++;
}

printf("Number of words: %d\n", word_count);

for (int i=0; i<word_count; i++)
{
    fprintf(fout, "wordList[%2d]: %s (length: %d)\n",
        i, wordList[i], strlen(wordList[i]));
}
fprintf(fout, "\n");

fprintf(fout, "File writing with fprintfStringArray(): \n");
fprintfStringArray(fout, wordList, word_count, 10);
fclose(fin);
fclose(fout);

```

```

void fprintStringArray(FILE *fout, char wordList[][MAX_WORD_LEN],
    int num_words, int words_per_line)
{
    int count = 0;
    char word[MAX_WORD_LEN];

    while (count < num_words)
    {
        for (int i = 0; i < words_per_line; i++)
        {
            strcpy(word, wordList[count]);
            fprintf(fout, "%-15s", word);
            printf("%-15s", word);
            count++;
            if (count >= num_words)
            {
                fprintf(fout, "\n");
                printf("\n");
                return;
            }
        }
        fprintf(fout, "\n");
        printf("\n");
    }
}

```



◆ input.txt

one two three four five six seven eight nine ten
January February March April May June July August September October
Sunday Monday Tuesday Wednesday Thursday Friday Saturday week day month
China India UnitedStates Indonesia Brazil Pakistan Nigeria Russia Bangladesh
Japan Mexico Philippines Vietnam Ethiopia Germany Egypt Iran Turkey Congo
Thailand France UnitedKingdom Italy



◆ output.txt

```
1 wordList[ 0]: one (length: 3)
2 wordList[ 1]: two (length: 3)
3 wordList[ 2]: three (length: 5)
4 wordList[ 3]: four (length: 4)
5 wordList[ 4]: five (length: 4)
6 wordList[ 5]: six (length: 3)
7 wordList[ 6]: seven (length: 5)
8 wordList[ 7]: eight (length: 5)
9 wordList[ 8]: nine (length: 4)
10 wordList[ 9]: ten (length: 3)
```

.....

```
46 wordList[45]: Egypt (length: 5)
47 wordList[46]: Iran (length: 4)
48 wordList[47]: Turkey (length: 6)
49 wordList[48]: Congo (length: 5)
50 wordList[49]: Thailand (length: 8)
51 wordList[50]: France (length: 6)
52 wordList[51]: UnitedKingdom (length: 13)
53 wordList[52]: Italy (length: 5)
```

```
54
```

```
55 File writing with fprintfStringArray():
```

```
56 one         two         three        four         five         six         seven        eight        nine        ten
57 January     February    March       April         May           June        July         August      September   October
58 Sunday      Monday      Tuesday     Wednesday    Thursday     Friday      Saturday    week        day         month
59 China       India       UnitedStates Indonesia    Brazil       Pakistan    Nigeria     Russia     Bangladesh  Japan
60 Mexico     Philippines Vietnam     Ethiopia     Germany     Egypt       Iran         Turkey     Congo       Thailand
61 France     UnitedKingdom Italy
```



Homework 5

Homework 5

5.1 다중 소스파일 프로그램의 장점에 대하여 설명하라. (10점)

5.2 정수형 난수 배열 관련 함수들을 위한 BigArray.cpp, BigArray.h 구현 (20점)

- 1) 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.
- 2) 주어진 배열에 포함되어 있는 정수 원소들의 위치를 뒤섞는 void shuffleBigArray(int *array, int size)를 BigArray.cpp 소스코드 파일에 구현하라.
- 3) 정수 배열의 내용을 화면으로 출력하는 함수 printBigArraySample(int *array, int size, int line_size=10, int num_sample_lines = 2)를 BigArray.cpp 파일에 구현하라. 이 함수는 주어진 배열을 한 줄에 line_size 개수의 원소를 출력한다. 배열의 크기가 50을 초과하면 맨 첫 2줄과 맨 끝 2줄을 출력하고, 중간에 "... " 을 표시 할 것. line_size가 별도로 지정되지 않는 경우, 기본값 10으로 설정할 것.
- 4) 정수 배열의 내용을 지정된 파일에 출력하는 함수 fprintfBigArraySample(FILE *fout, int *array, int size, int line_size=10, int num_sample_lines = 2)를 BigArray.cpp에 구현하라. 출력 방법은 위 printBigArraySample()과 동일하게 할 것.
- 5) 위에서 구현한 배열 관련 함수들의 함수 원형을 포함하는 헤더파일 BigArray.h를 구현하라.



Homework 5

5.2 정수형 난수 배열 관련 함수들을 위한 BigArray.cpp, BigArray.h 구현 (계속)

- 6) main() 함수에서는 BigArray.h 헤더 파일을 포함시켜 정수형 배열 관련 함수를 사용할 수 있도록 구성하라.
- 7) main() 함수에서 정수형 배열 (배열의 크기는 임의로 정의할 것)을 선언하고, genBigRandArray() 함수를 호출하여 이 정수형 배열에 중복되지 않는 난수를 생성하고, fprintfBigArraySample() 함수를 사용하여 출력 파일 "output.txt"에 출력하라.
- 8) 전체 프로그램을 다중 소스파일로 구현하라.
- 9) 출력 결과 (화면 출력 및 파일 저장) 예시:

```
Input size of big array (more than 100000) = 1000000
361955  579012  145516  140462  515280  969585  888635  402539  152102  176572
402892  860889  580638  870044  139319  509022  27666  263304  82555  212774
563715  389629  113528  147999  801172  151740  301173  585275  845005  71050

517897  852209  807221  219050  427238  145368  733640  155275  6680  200430
781684  862898  970663  245634  819539  6740  391286  298149  999958  42057
819435  698914  659016  697411  542744  531948  431882  407191  875159  6546
```



Homework 5

5.2 정수형 난수 배열 관련 함수들을 위한 BigArray.cpp, BigArray.h 구현 (계속, Bonus)

10) 동적메모리 할당을 사용하여 100만개 이상의 정수 배열을 생성하고, 중복되지 않는 난수를 생성하라.

11) 출력 결과 (화면 출력 및 파일 저장) 예시:

```
Input size of big array (more than 100000) = 1000000
361955 579012 145516 140462 515280 969585 888635 402539 152102 176572
402892 860889 580638 870044 139319 509022 27666 263304 82555 212774
563715 389629 113528 147999 801172 151740 301173 585275 845005 71050
```

```
517897 852209 807221 219050 427238 145368 733640 155275 6680 200430
781684 862898 970663 245634 819539 6740 391286 298149 999958 42057
819435 698914 659016 697411 542744 531948 431882 407191 875159 6546
```

```
Input size of big array (more than 100000) = 10000000
5582469 9061014 9324359 9030755 7817314 6510701 3805326 2625101 5032138 7347221
6034895 9033824 5589826 3981485 3888557 1580909 8825648 6864474 8890228 1744011
2828513 2173125 8779793 8382869 1191812 6388507 1894165 3355177 9566124 5170957
```

```
5452512 2446016 8483565 3223882 4861759 6023237 2774862 3931130 4309847 7470631
5513744 7077893 3781048 2569534 48207 3817045 6340556 9313852 811972 6210752
8597086 2692991 5644388 7916064 7507867 3827334 5081792 8457052 8311918 8322102
```

