Fundamentals of Computer System
- chapter 13. File input and output

민기복

Ki-Bok Min, PhD

서울대학교 에너지자원공학과 조교수
Assistant Professor, Energy Resources Engineering

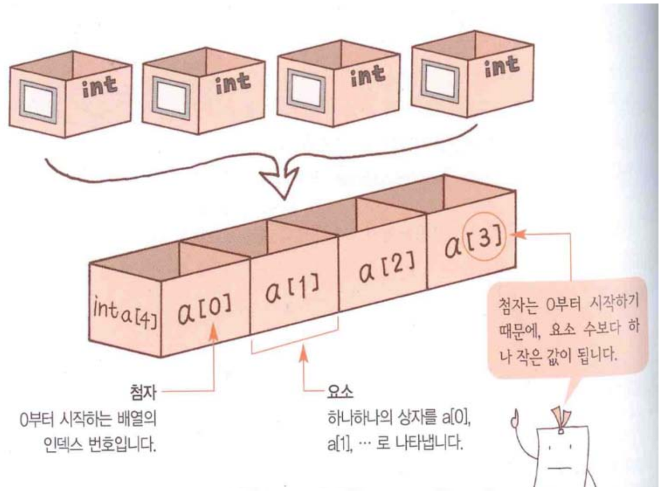
SEOUL NATIONAL UNIVERSITY

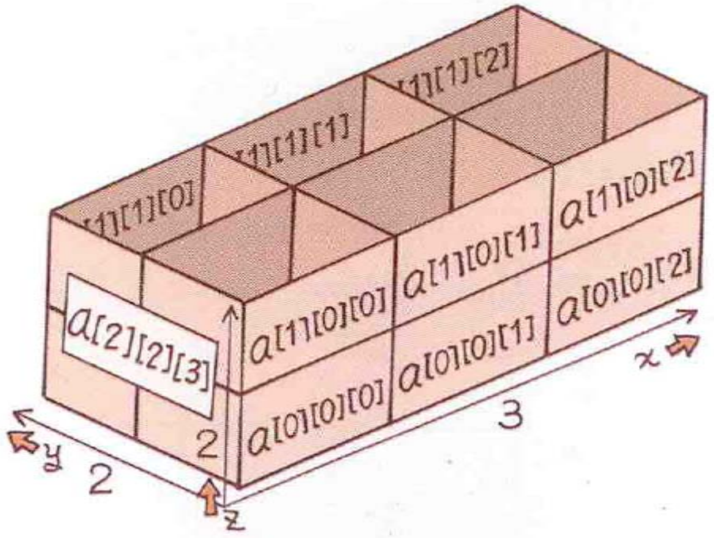
Last Week

Chapter 10. Arrays and Pointers (배열과 포인터)

SEOUL NATIONAL UNIVERSITY

- Arrays – initialization, assignment,
- Multidimensional arrays
- Pointers and arrays
- Functions, Arrays, and Pointers
- Pointer operations


int
int
int
int
int a[4]
a[0]
a[1]
a[2]
a[3]
첨자는 0부터 시작하기 때문에, 요소 수보다 하나 작은 값이 됩니다.
첨자
0부터 시작하는 배열의 인덱스 번호입니다.
요소
하나하나의 상자를 a[0], a[1], … 로 나타냅니다.



Pointers: a first look
What is it?

SEOUL NATIONAL UNIVERSITY

- Pointer: a variable whose value is a memory address
- 포인터는 주소를 값으로 가지는 변수이다.
  - Char 형 변수 → 문자
  - int형 변수 → 정수
  - 포인터변수 →
- People: a name and a value
- Computer: an address (a computer’s version of name) and a value

Pointers: a first look
the indirection operator(간접연산자)

SEOUL NATIONAL UNIVERSITY

```
p = &a;
b = *p;
```

←→

```
b = a
```

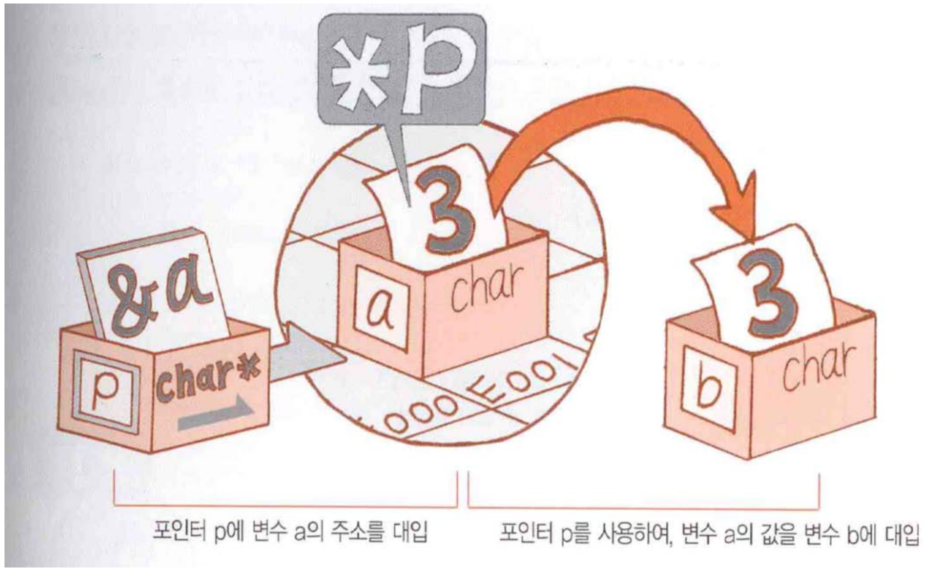
p가 가리키는 주소의 값

ex)

a = 3;

p = &a;   //a를 가리키는 포인터

b = *p;   //p이 가리키고 있는주소의 값을 b에 대입


*p
3
&a
p
char*
a
char
000
1003
3
b
char
포인터 p에 변수 a의 주소를 대입
포인터 p를 사용하여, 변수 a의 값을 변수 b에 대입

Pointers
Pointers and Arrays (포인터와 배열)

- Array is simply a disguised use of pointers. (배열표기는 실제로는 포인터의 변장된 사용에 불과하다)
- 배열명은 곧 그 배열의 시작주소이다.
- Pointers can do array subscripting operations.
- For an array flizny, the following is true.
  - flizny == &flizny[0]
  - Both flizny and &flizny[0] represent the memory address of first element. Both are constants because they remain fixed.

Pointers
pointers and arrays

SEOUL NATIONAL UNIVERSITY

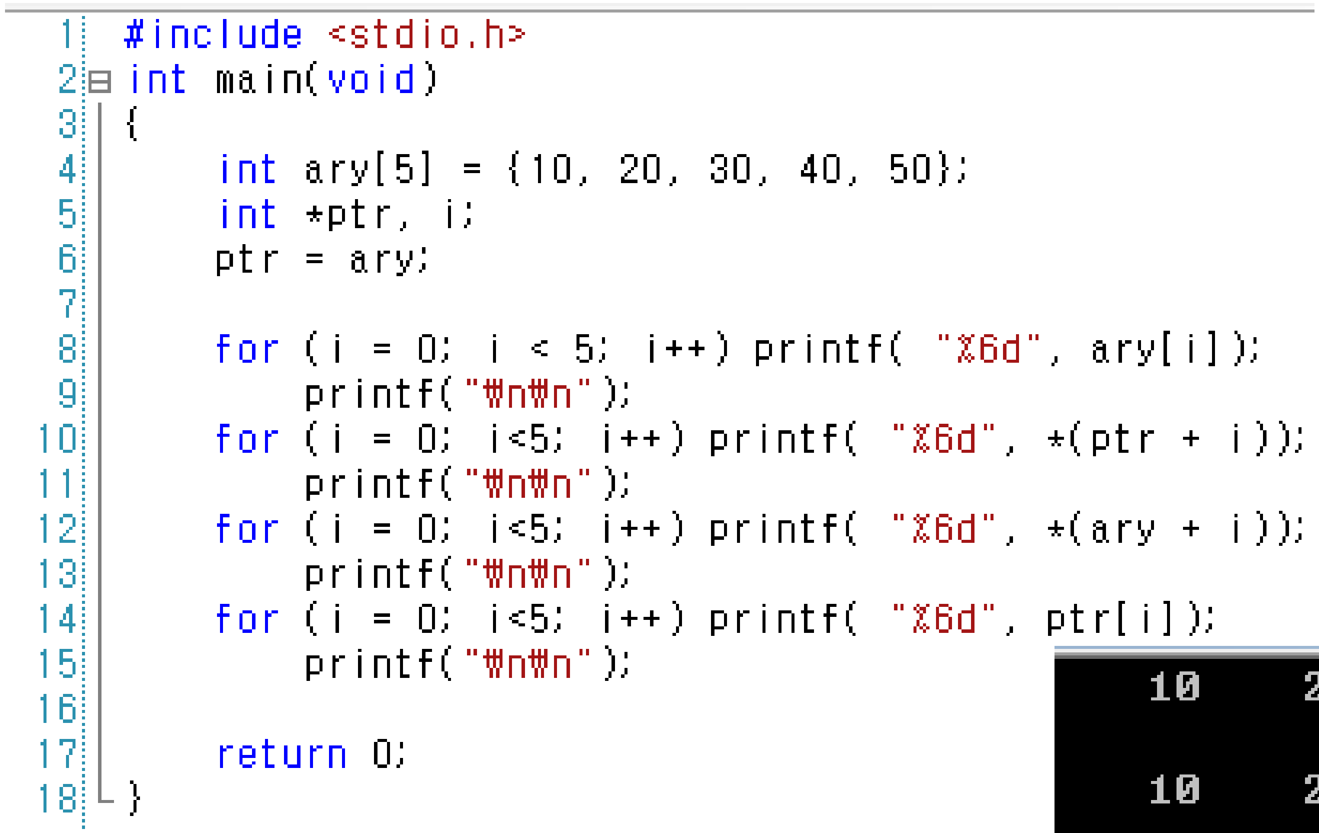
- With an array *dates*,

  dates + 2 == &dates[2] /* 주소가 같다.*/

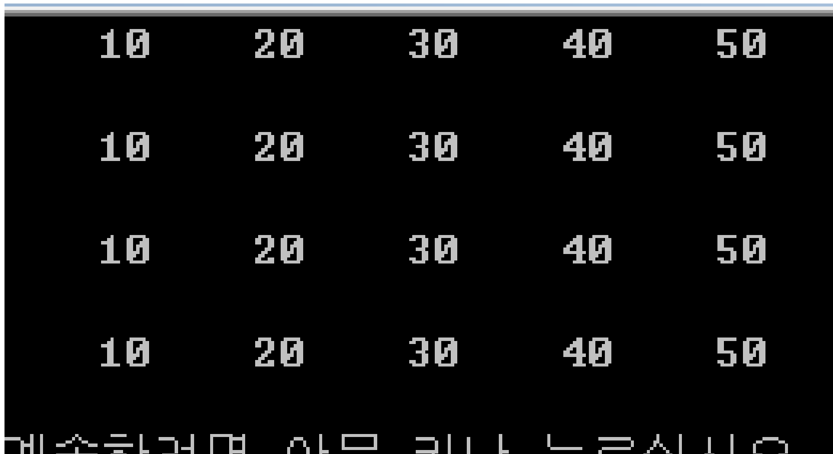
  (dates + 2) == dates[2] / 값이 같다.*/

- Close connection between arrays and pointers!!!
- Use a pointer to identify an individual element of an array and to obtain its value.
- Two different notations for the same thing.
- 실제로 C언어 표준은 배열표기를 포인터로 서술한다. ar[n] → *(ar+n)

Pointers
Pointers and Arrays (포인터와 배열)



```c
#include <stdio.h>
int main(void)
{
	int ary[5] = {10, 20, 30, 40, 50};
	int *ptr, i;
	ptr = ary;

	for (i = 0; i < 5; i++) printf( "%6d", ary[i]);
		printf("₩n₩n");
	for (i = 0; i<5; i++) printf( "%6d", *(ptr + i));
		printf("₩n₩n");
	for (i = 0; i<5; i++) printf( "%6d", *(ary + i));
		printf("₩n₩n");
	for (i = 0; i<5; i++) printf( "%6d", ptr[i]);
		printf("₩n₩n");

	return 0;
}
```



```
    10    20    30    40    50

    10    20    30    40    50

    10    20    30    40    50

    10    20    30    40    50

```

pointer2.c

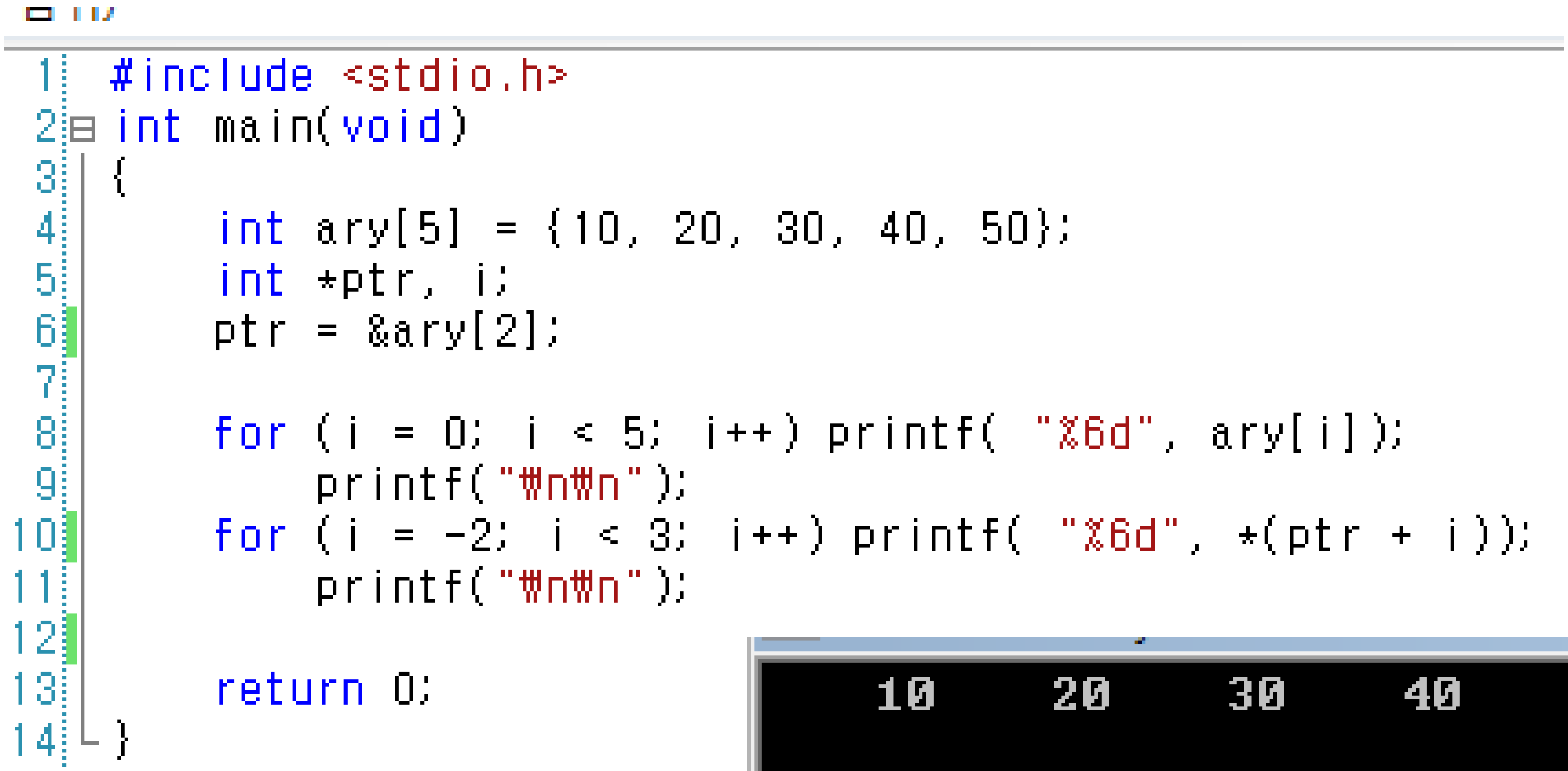
Pointers
pointers and arrays

SEOUL NATIONAL UNIVERSITY

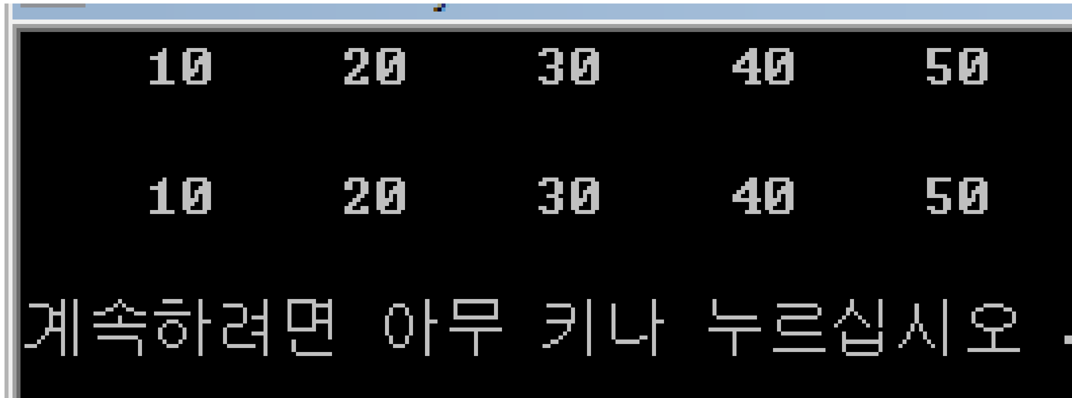
- 포인터에 1을 더하면 C는 하나의 기억단위를 더한다 (short – 2 byte,int – 4 byte, double – 8 byte).
- 즉, 주소가 다음 바이트가 아니라 다음 원소(element)의 주소로 증가한다 – 포인터가 가리키는 객체의 종류를 선언해야 하는 이유.
- 데이터 객체(data object)는 값을 저장하는 데 사용할 수 있는 데이터 저장 영역을 일반적으로 지칭하는 용어

p가 포인터 변수일때 p+i가 의미하는 값은
p + i* sizeof(대상체) 이다.

Pointers
pointers and arrays



```
#include <stdio.h>
int main(void)
{
	int ary[5] = {10, 20, 30, 40, 50};
	int *ptr, i;
	ptr = &ary[2];

	for (i = 0; i < 5; i++) printf( "%6d", ary[i]);
		printf("₩n₩n");
	for (i = -2; i < 3; i++) printf( "%6d", *(ptr + i));
		printf("₩n₩n");

	return 0;
}
```



```
    10    20    30    40    50

    10    20    30    40    50

계속하려면 아무 키나 누르십시오
```

pointer3.c

Pointers
Functions that operates on an array

SEOUL NATIONAL UNIVERSITY

- Suppose you want a function that returns the sum of the elements of an array, *marbles*

- Calling a function
  - total = sum(marbles);   // 가능한 함수 호출의 예
- Prototype (declaration)
  - int sum(int * ar)   // 대응하는 함수 프로토타입

배열이름은 첫 번째 원소의 주소이기 때문에 배열 이름을 실전달인자로 사용하려면 대응하는 형식매개변수가 포인터여야 한다.

Pointers
Functions that operates on an array

SEOUL NATIONAL UNIVERSITY

- Definition

```
int sum(int * ar)
{
    int i;
    int total = 0;
    for (i = 0; i < 10; i++)    // 원소가 10개라고 가정
        total += ar[ i ];        // ar[ i ]는 *(ar + i)와 같다.
    return  total;
}
```

형식매개변수의 선언에 사용될때 다음과 같이 써도된다.
int ar[ ]

Use array notation with a pointer
포인터에 배열표기를 사용

Pointers
Usinig Pointer Parameters

SEOUL NATIONAL UNIVERSITY

```
int sum(int * ar, int n);
int sum(int *   , int   );
int sum(int  ar[ ], int n);
int sum(int  [ ], int) ;
```

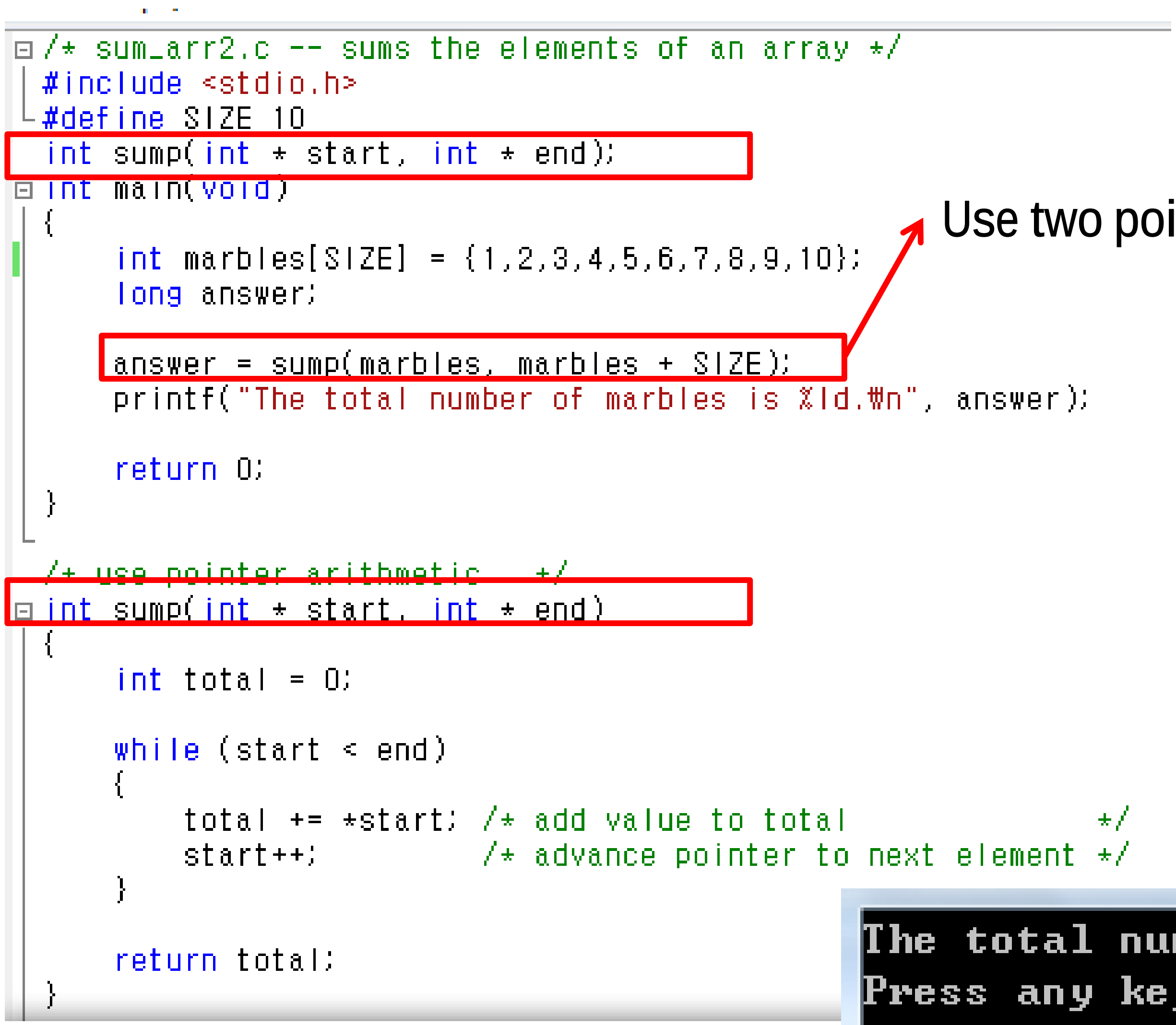
Four prototypes are identical

```
int sum(int * ar, int n)
{ …}

int sum(int ar[ ], int n)
{ …}
```

Two definitions are identical

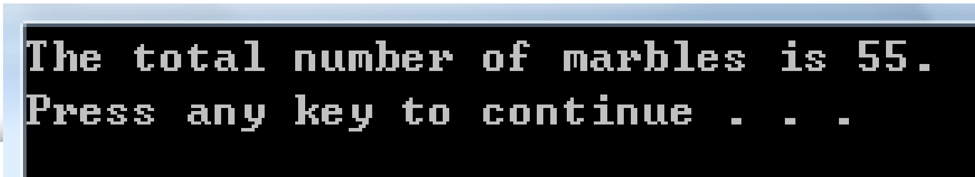
Pointers
Using Pointer Parameters

SEOUL NATIONAL UNIVERSITY



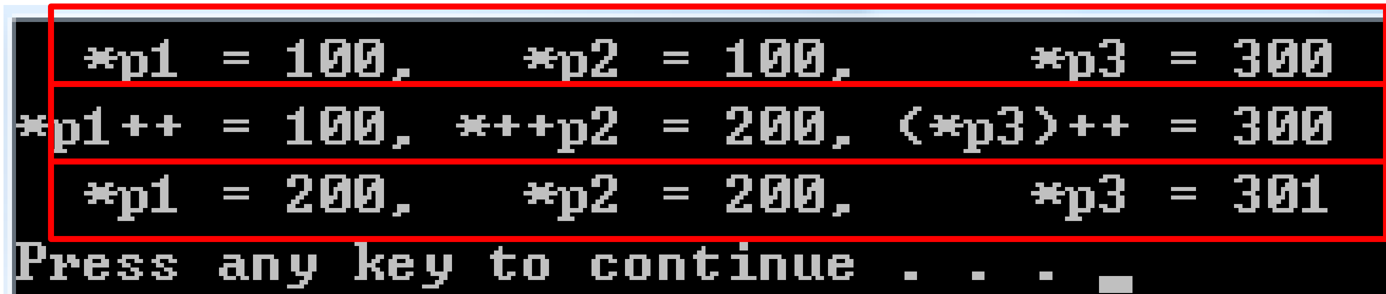
```
/* sum_arr2.c -- sums the elements of an array */
#include <stdio.h>
#define SIZE 10
int sump( int * start, int * end);
int main(void)
{
    int marbles[SIZE] = {1,2,3,4,5,6,7,8,9,10};
    long answer;

    answer = sump(marbles, marbles + SIZE);
    printf("The total number of marbles is %ld.₩n", answer);

    return 0;
}

/* use pointer arithmetic    */
int sump( int * start, int * end)
{
    int total = 0;

    while (start < end)
    {
        total += *start;  /* add value to total              */
        start++;          /* advance pointer to next element */
    }

    return total;
}
```

Use two pointers to describe the array



```
The total number of marbles is 55.
Press any key to continue . . .
```

```c
/* order.c -- precedence in pointer operations */
#include <stdio.h>
int data[2] = {100, 200};
int moredata[2] = {300, 400};
int main(void)
{
      int * p1, * p2, * p3;

      p1 = p2 = data;
      p3 = moredata;
      printf("  *p1 = %d,   *p2 = %d,     *p3 = %d₩n",
                *p1     ,   *p2     ,     *p3);
      printf("*p1++ = %d, *++p2 = %d, (*p3)++ = %d₩n",
              *p1++     , *++p2     , (*p3)++);
      printf("  *p1 = %d,   *p2 = %d,     *p3 = %d₩n",
                *p1     ,   *p2     ,     *p3);

      return 0;
}
```



```
  *p1 = 100,   *p2 = 100,     *p3 = 300
*p1++ = 100, *++p2 = 200, (*p3)++ = 300
  *p1 = 200,   *p2 = 200,     *p3 = 301
Press any key to continue . . .
```

Pointers
Pointer operations

SEOUL NATIONAL UNIVERSITY

- Assignment
  - ptr1 = urn;   //assign an address to a pointer
- Value finding
- Taking a pointer address
- Adding an integer to a pointer
- Incrementing a pointer
- Subtracting an integer from a pointer
- Decrementing a pointer

Pointers
Pointer operations

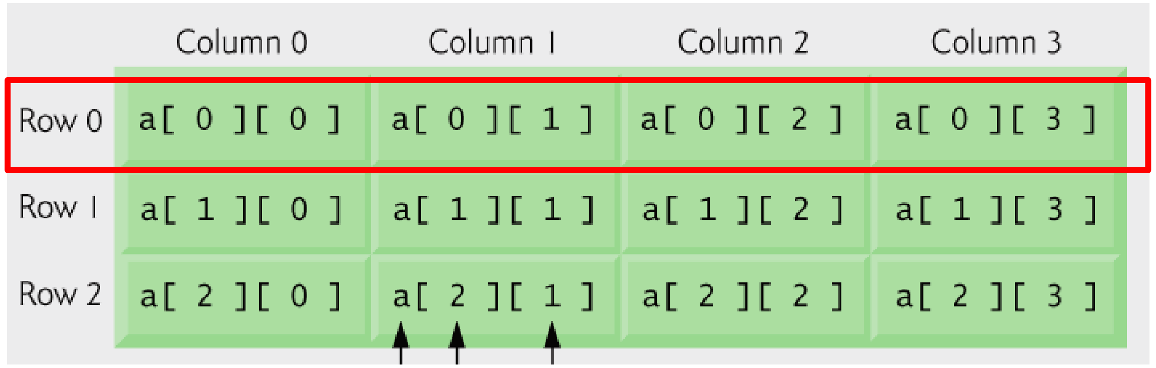
SEOUL NATIONAL UNIVERSITY

- Differencing
- Comparisons

Pointers
Pointers and multidimensional arrays

SEOUL NATIONAL UNIVERSITY

- int zippo[3][4]
- zippo는 그 배열의 첫번째 원소의 주소
  → zippo ==&zippo[0]
  → zippo[0] == &zippo[0][0]
  →*(zippo[0]) ==zippo[0][0]
  →*(zippo) == zippo[0]
- **zippo = zippo[0][0]



Pointers
Pointers and multidimensional arrays

- 값 표현

  zippo[m][n] == *(*(zippo+ m) + n)

- 주소 표현

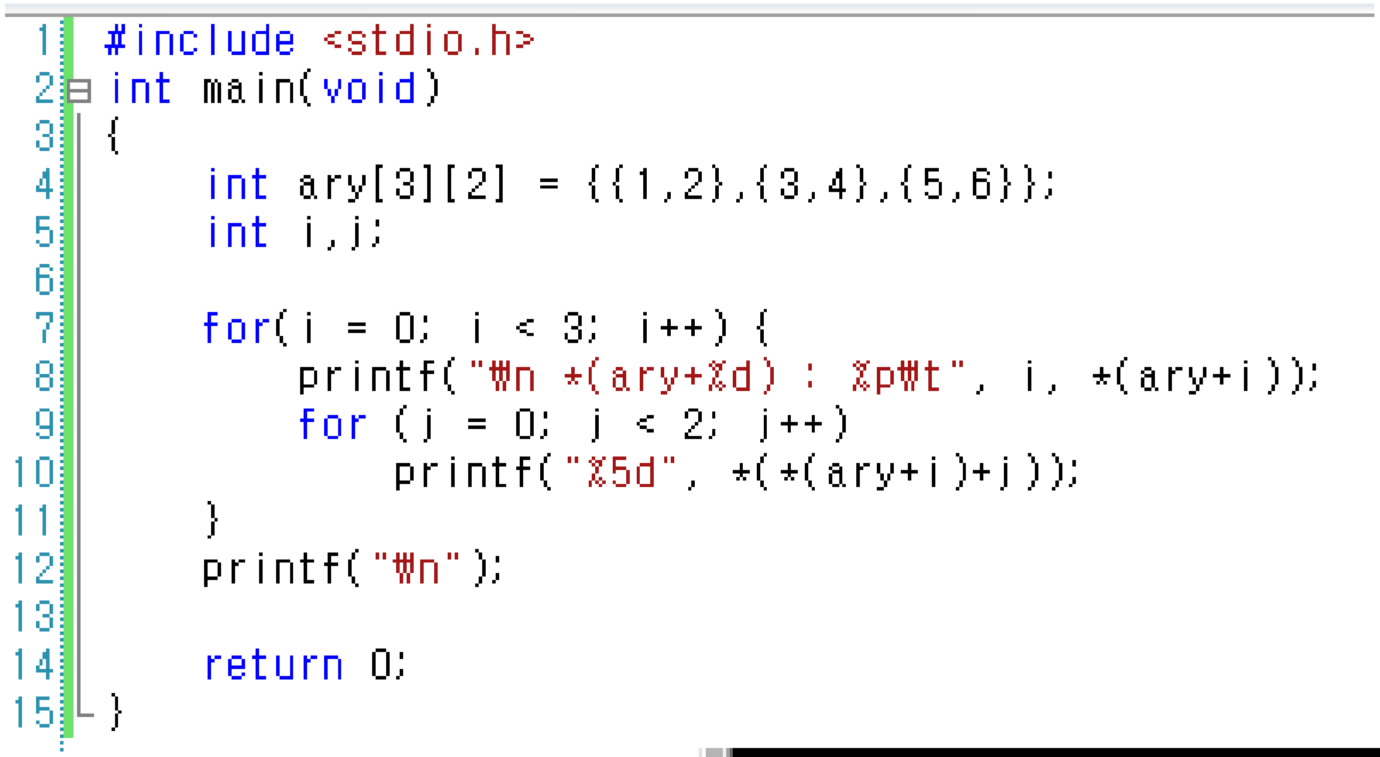
  &zippo[m][n] == (*(zippo+m) + n)

Compiler use these form → faster computation.

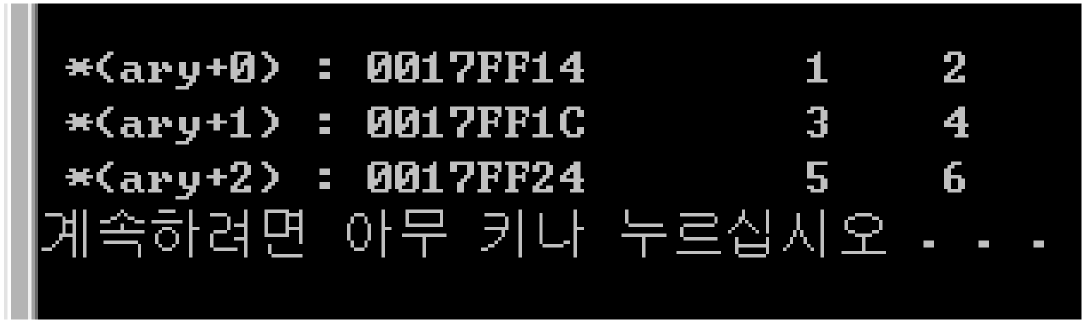
- 2차원 배열을 나타내는 포인터 변수에는 *를 두 개를 써야 비로소 값을 나타낸다. *를 하나 썼을 때 여전히 주소를 나타냄 → 2차원 포인터.
- 1차원 포인터 → * zippo → 값
- 2차원 포인터 → ** zippo → 값

Pointers
Pointers and multidimensional arrays

SEOUL NATIONAL UNIVERSITY

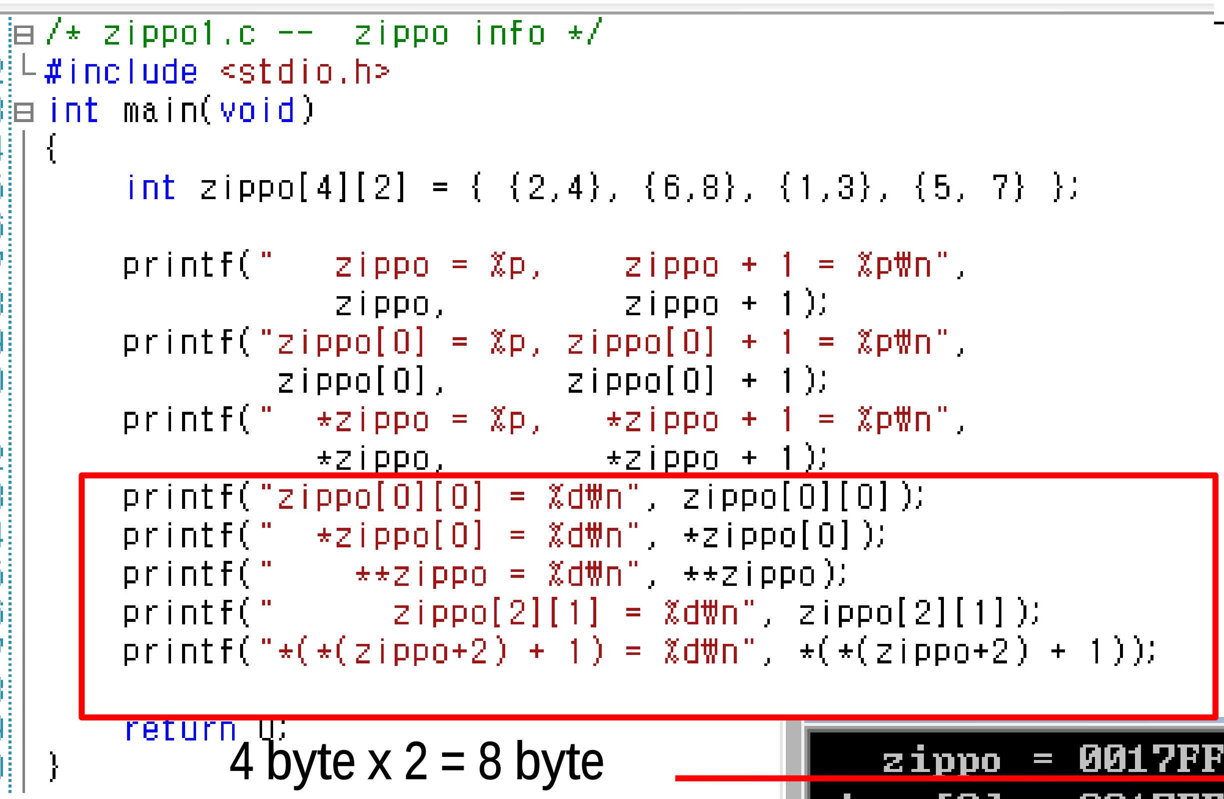


```c
#include <stdio.h>
int main(void)
{
	int ary[3][2] = {{1,2},{3,4},{5,6}};
	int i,j;

	for(i = 0; i < 3; i++) {
		printf("₩n *(ary+%d) : %p₩t", i, *(ary+i));
		for (j = 0; j < 2; j++)
			printf("%5d", *(*(ary+i)+j));
	}
	printf("₩n");

	return 0;
}
```



```
*(ary+0) : 0017FF14        1    2
*(ary+1) : 0017FF1C        3    4
*(ary+2) : 0017FF24        5    6
계속하려면 아무 키나 누르십시오 . . .
```

Pointers
Pointers and multidimensional arrays

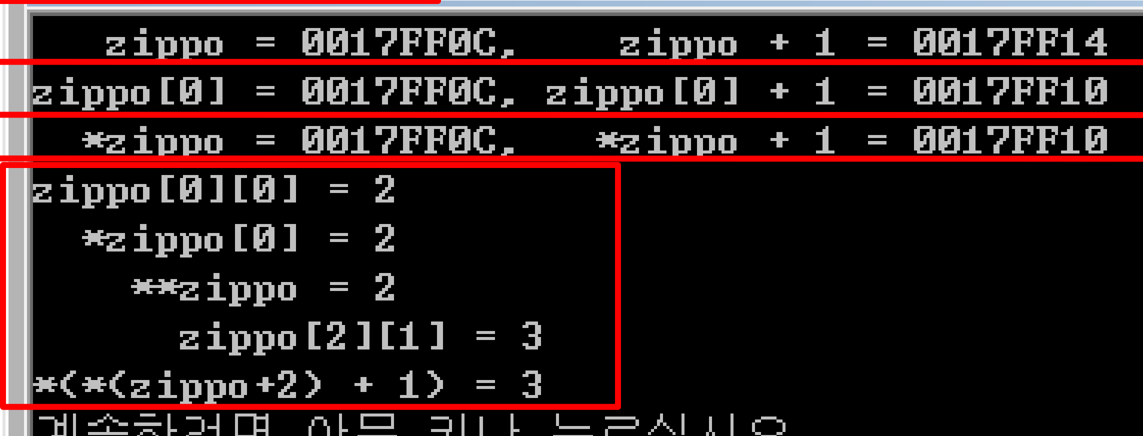


```
/* zippo1.c --  zippo info */
#include <stdio.h>
int main(void)
{
    int zippo[4][2] = { {2,4}, {6,8}, {1,3}, {5, 7} };

    printf("   zippo = %p,    zippo + 1 = %p₩n",
               zippo,         zippo + 1);
    printf("zippo[0] = %p, zippo[0] + 1 = %p₩n",
            zippo[0],      zippo[0] + 1);
    printf("  *zippo = %p,   *zippo + 1 = %p₩n",
              *zippo,        *zippo + 1);
    printf("zippo[0][0] = %d₩n", zippo[0][0]);
    printf("  *zippo[0] = %d₩n", *zippo[0]);
    printf("    **zippo = %d₩n", **zippo);
    printf("      zippo[2][1] = %d₩n", zippo[2][1]);
    printf("*(*(zippo+2) + 1) = %d₩n", *(*(zippo+2) + 1));

    return 0;
}
```

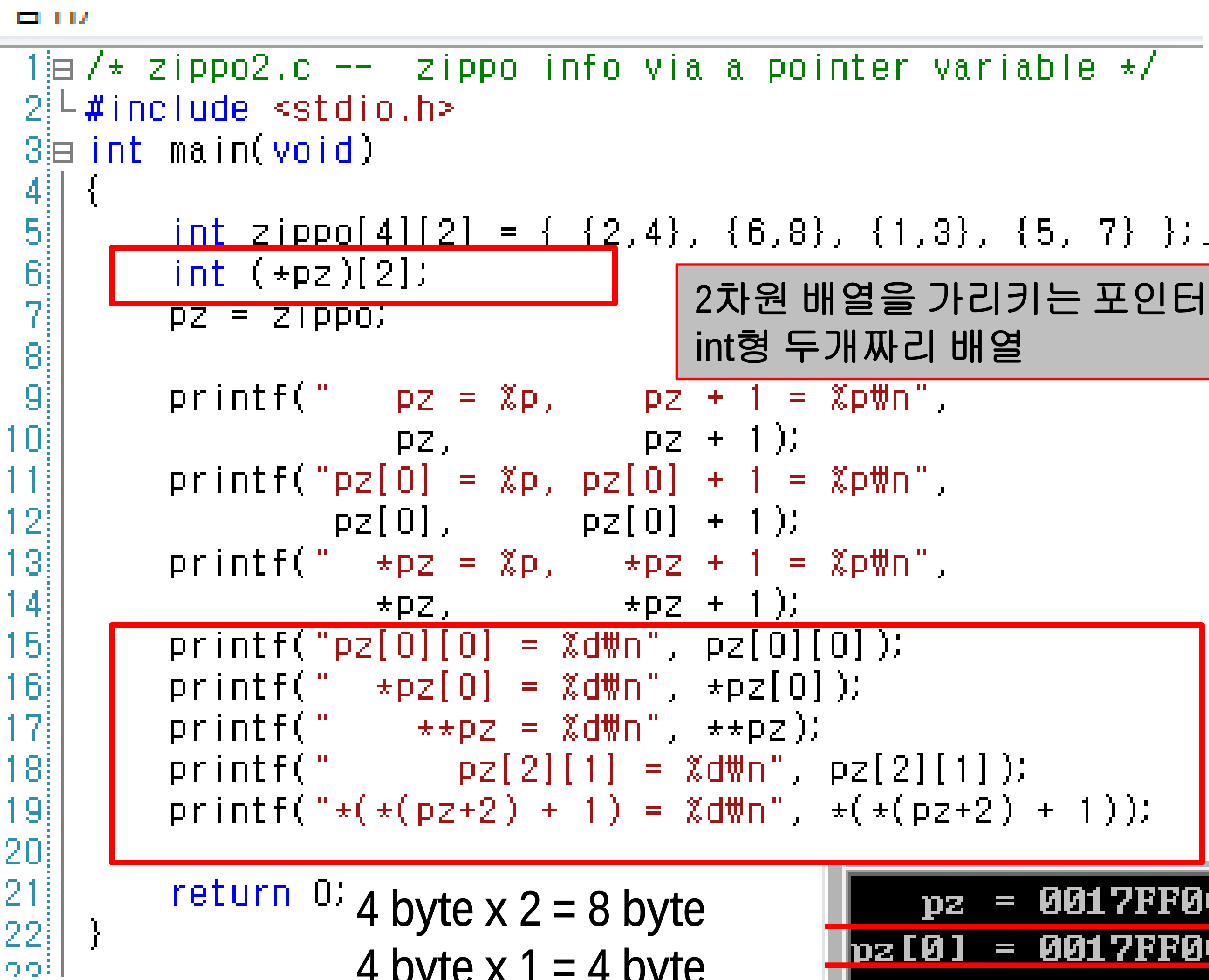
4 byte x 2 = 8 byte

4 byte x 1 = 4 byte

4 byte x 1 = 4 byte



```
   zippo = 0017FF0C,    zippo + 1 = 0017FF14
zippo[0] = 0017FF0C, zippo[0] + 1 = 0017FF10
  *zippo = 0017FF0C,   *zippo + 1 = 0017FF10
zippo[0][0] = 2
  *zippo[0] = 2
    **zippo = 2
      zippo[2][1] = 3
*(*(zippo+2) + 1) = 3
계속하려면 아무 키나 누르십시오
```



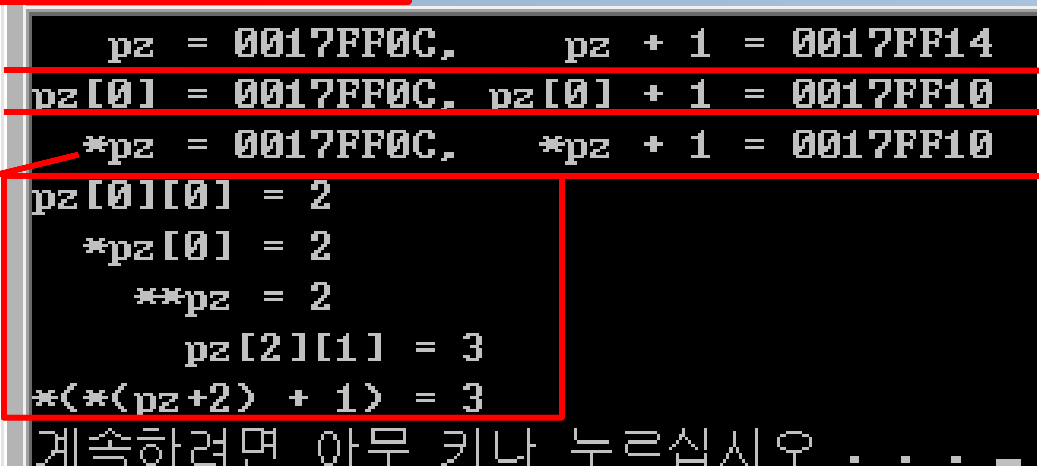
```c
/* zippo2.c --  zippo info via a pointer variable */
#include <stdio.h>
int main(void)
{
        int zippo[4][2] = { {2,4}, {6,8}, {1,3}, {5, 7} };
        int (*pz)[2];
        pz = zippo;

        printf("   pz = %p,     pz + 1 = %p₩n",
                   pz,          pz + 1);
        printf("pz[0] = %p, pz[0] + 1 = %p₩n",
                pz[0],       pz[0] + 1);
        printf("  *pz = %p,    *pz + 1 = %p₩n",
                  *pz,         *pz + 1);
        printf("pz[0][0] = %d₩n", pz[0][0]);
        printf("  *pz[0] = %d₩n", *pz[0]);
        printf("    **pz = %d₩n", **pz);
        printf("      pz[2][1] = %d₩n", pz[2][1]);
        printf("*(*(pz+2) + 1) = %d₩n", *(*(pz+2) + 1));

        return 0;
}
```

2차원 배열을 가리키는 포인터 변수의 선언
int형 두개짜리 배열

4 byte x 2 = 8 byte

4 byte x 1 = 4 byte

4 byte x 1 = 4 byte

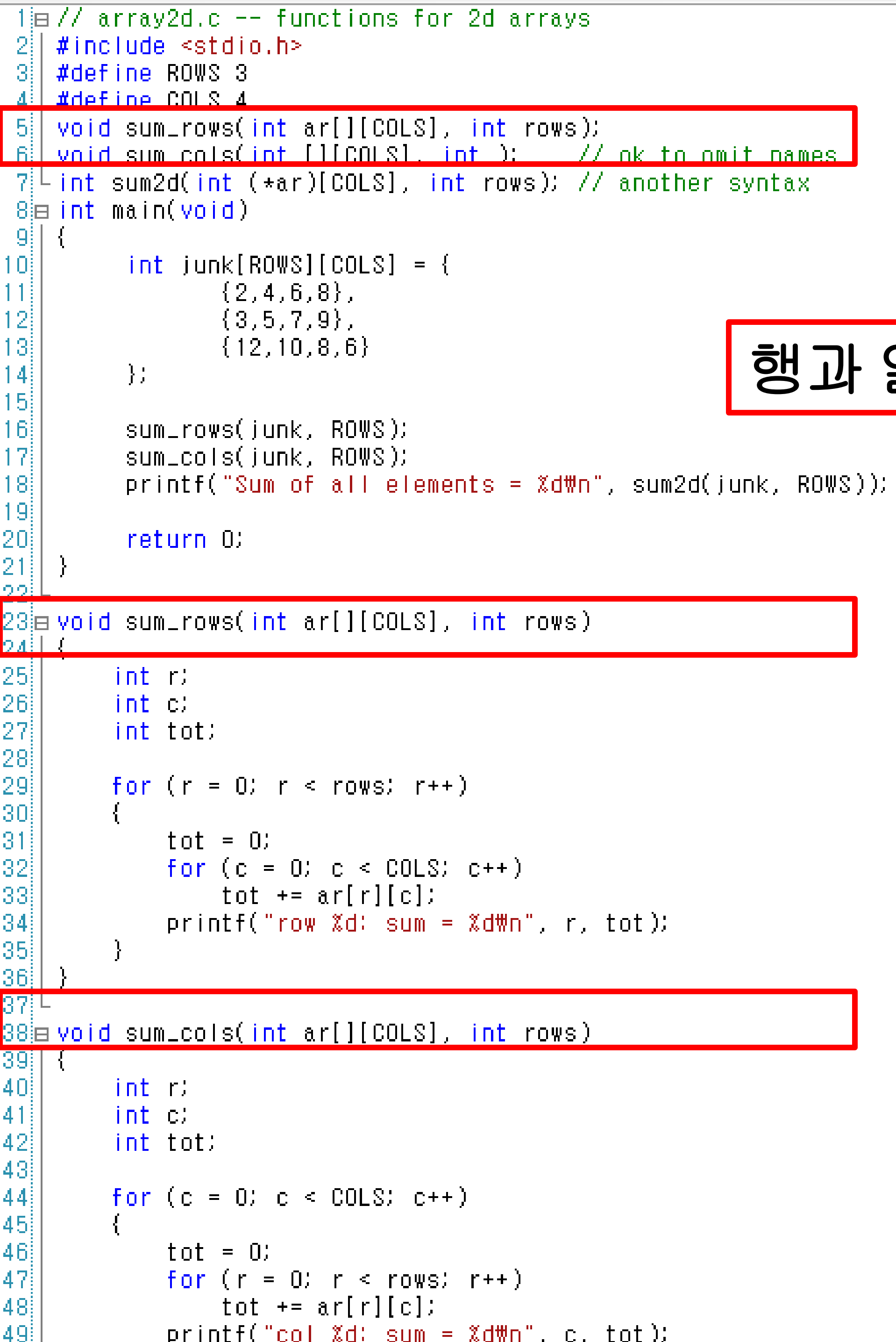


```
   pz = 0017FF0C,     pz + 1 = 0017FF14
pz[0] = 0017FF0C, pz[0] + 1 = 0017FF10
  *pz = 0017FF0C,    *pz + 1 = 0017FF10
pz[0][0] = 2
  *pz[0] = 2
    **pz = 2
      pz[2][1] = 3
*(*(pz+2) + 1) = 3
계속하려면 아무 키나 누르십시오 . . .
```

Pointers
Functions and multidimensional arrays

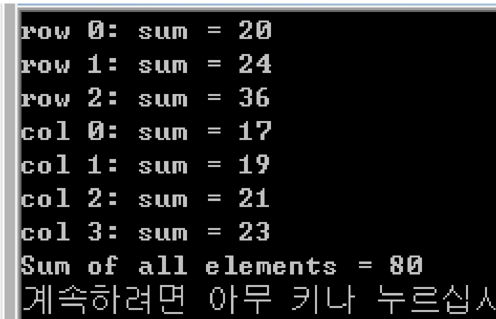
SEOUL NATIONAL UNIVERSITY

- 2차원 배열을 전달인자로 하는 함수의 예 (pt가 형식매개변수일때)
- Int junk[3][4] = {{2,4,5,8},{3,5,6,9},{12,10,8,6}};
- …
- void somefunction (int (* pt)[4]); or
- void somefunction(int pt[ ][4]);



```c
// array2d.c -- functions for 2d arrays
#include <stdio.h>
#define ROWS 3
#define COLS 4
void sum_rows(int ar[][COLS], int rows);
void sum_cols(int [][COLS], int );     // ok to omit names
int sum2d(int (*ar)[COLS], int rows); // another syntax
int main(void)
{
     int junk[ROWS][COLS] = {
            {2,4,6,8},
            {3,5,7,9},
            {12,10,8,6}
     };

     sum_rows(junk, ROWS);
     sum_cols(junk, ROWS);
     printf("Sum of all elements = %d\n", sum2d(junk, ROWS));

     return 0;
}

void sum_rows(int ar[][COLS], int rows)
{
    int r;
    int c;
    int tot;

    for (r = 0; r < rows; r++)
    {
        tot = 0;
        for (c = 0; c < COLS; c++)
            tot += ar[r][c];
        printf("row %d: sum = %d\n", r, tot);
    }
}

void sum_cols(int ar[][COLS], int rows)
{
    int r;
    int c;
    int tot;

    for (c = 0; c < COLS; c++)
    {
        tot = 0;
        for (r = 0; r < rows; r++)
            tot += ar[r][c];
        printf("col %d: sum = %d\n", c, tot);
```

행과 열의 합을 각각 구하는 프로그램



```
row 0: sum = 20
row 1: sum = 24
row 2: sum = 36
col 0: sum = 17
col 1: sum = 19
col 2: sum = 21
col 3: sum = 23
Sum of all elements = 80
계속하려면 아무 키나 누르십시
```

Chapter 10. Arrays and Pointers (배열과 포인터)

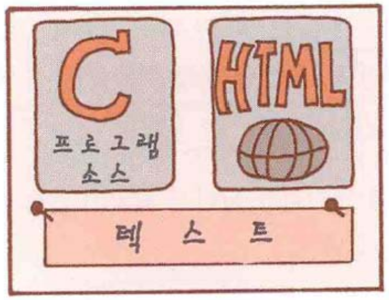
SEOUL NATIONAL UNIVERSITY

- Pointers and arrays
- Functions, Arrays, and Pointers
- Pointer operations
- Pointers and multidimensional (2D) arrays
- Functions and multidimensional (2D) arrays
- Try to understand the following files which are available in the eTL
  - pointer1.c, pointer2.c, pointer3.c, poiner4.c, order.c, zippo1.c, zippo2.c, array2d.c

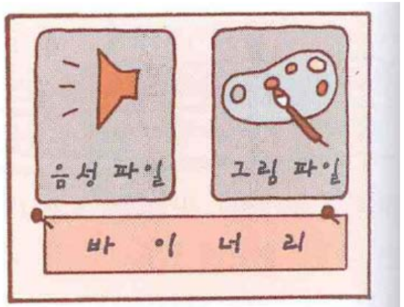
Chapter 13. File I/O
What is a file?

SEOUL NATIONAL UNIVERSITY

- File: 데이터나 프로그램 등을 디스크 상에 기록한 것
  - Text file : composed of text. we (human) can recognize it. Also called ASCII file
    - Ex) C source file
  - Binary file : composed of codes. Only programs can recognize it. we (human) have no idea.
    - Ex) object file



문자로서 읽을 수 있는 것
(C 언어의 프로그램 소스나 HTML 등)



문자로서 읽을 수 없는 것
(컴파일 후의 C 언어 프로그램, 영상 데이터 등)

File
High level Input/Output

SEOUL NATIONAL UNIVERSITY

- Low-level I/O: fundamental I/O services provided by the operating system
- Standard high-evel I/O: standard package of C library functions and stdio.h header file definition→ ANSI C.
  - Many specialized functions handling different I/O problems
  - Buffer (an intermediate storage area, 512 Byte ~ 16 MByte) is used for input & output. → greatly increase the data transfer rate

File
Order to handle a file

SEOUL NATIONAL UNIVERSITY

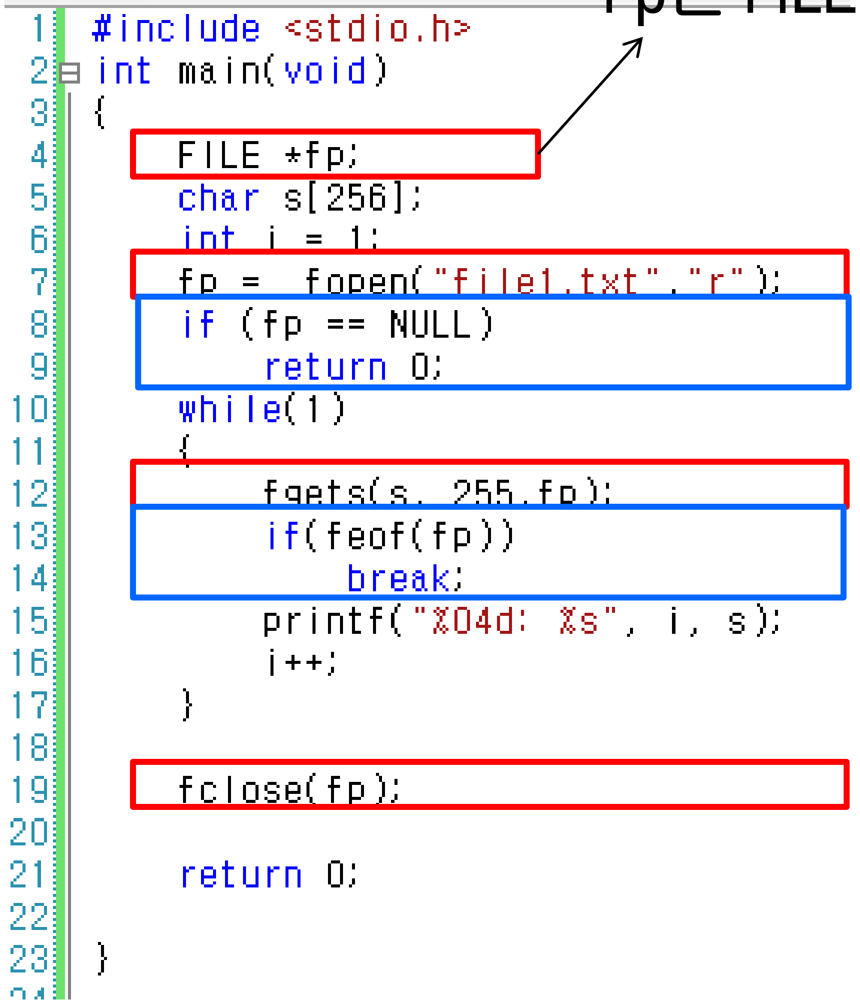
- Order to handle a file
  - Open a file and bring a file pointer
  - Read and write a file through a file pointer
  - Close a file

1 파일을 연다
FILE
2 읽기 쓰기를 수행한다
3 파일을 닫는다
FILE

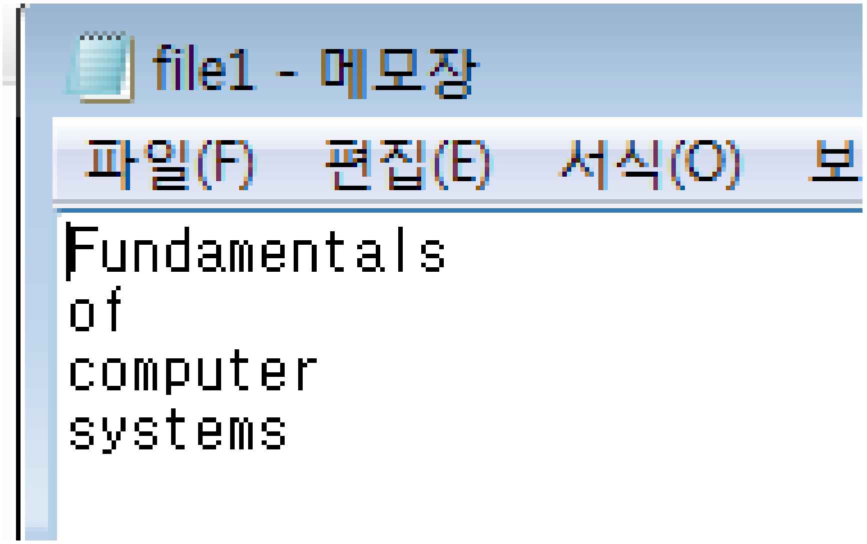
Reading a file
A simple example (read.c)

SEOUL NATIONAL UNIVERSITY

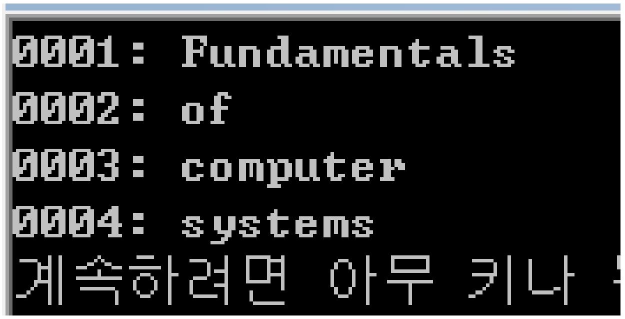
Fp는 FILE을 가리키는 포인터



```c
#include <stdio.h>
int main(void)
{
    FILE *fp;
    char s[256];
    int i = 1;
    fp =  fopen("file1.txt","r");
    if (fp == NULL)
        return 0;
    while(1)
    {
        fgets(s, 255,fp);
        if(feof(fp))
            break;
        printf("%04d: %s", i, s);
        i++;
    }

    fclose(fp);

    return 0;

}
```

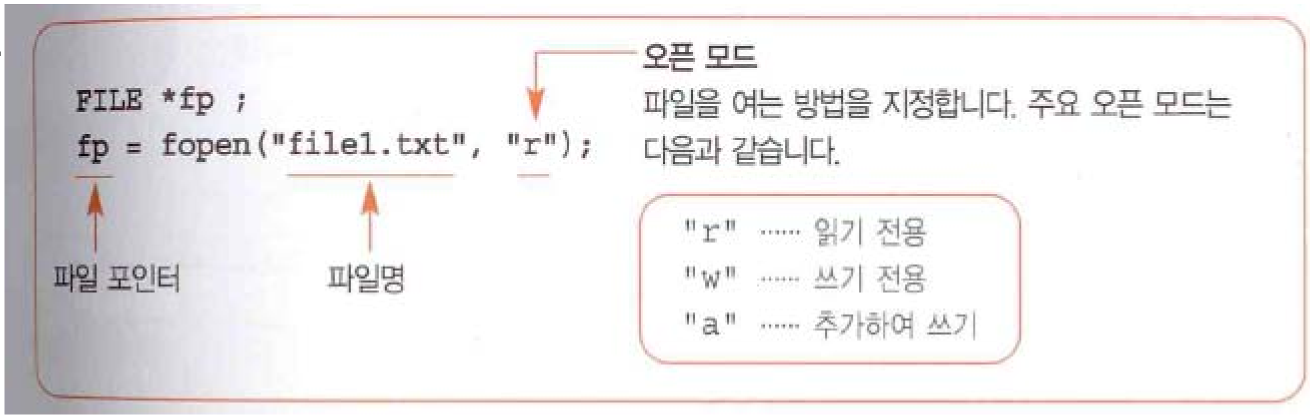


```
Fundamentals
of
computer
systems
```



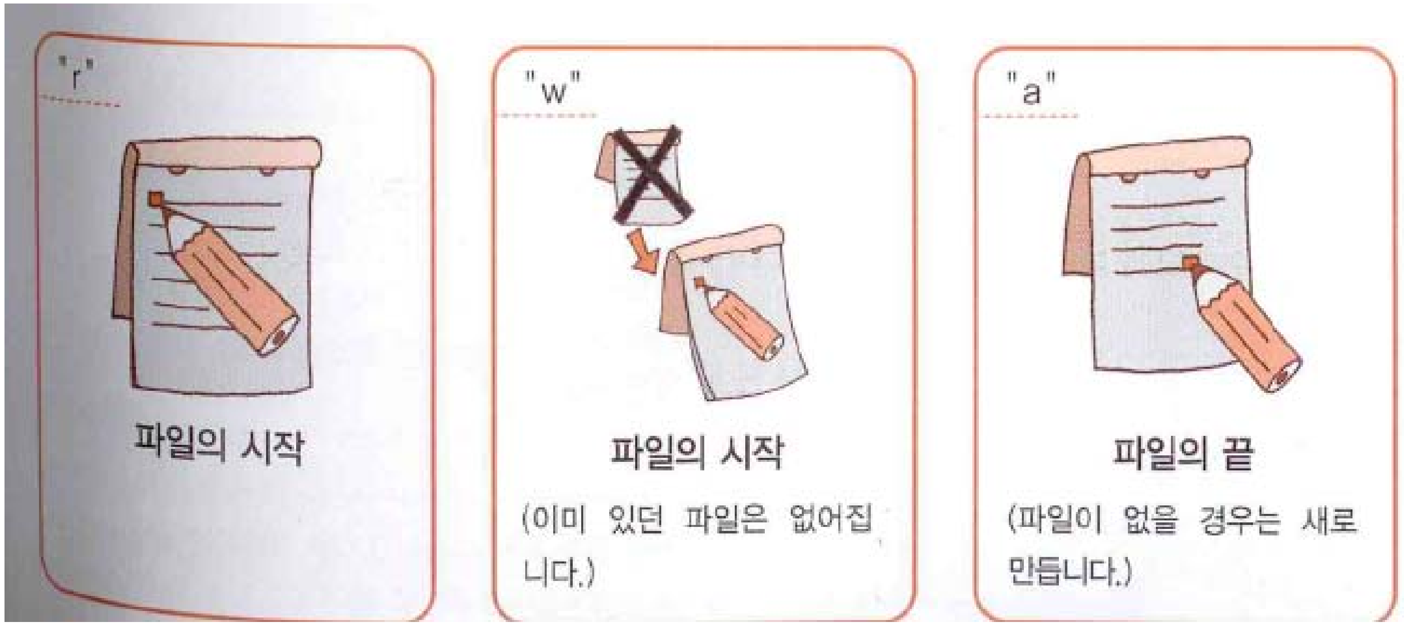
```
0001: Fundamentals
0002: of
0003: computer
0004: systems
계속하려면 아무 키나
```

fopen()

- fopen()
  - used to open a file.
  - Already declared in stdio.h
  - fopen() returns a file pointer (ex. fp). Returns NULL when it cannot open the file.
  - FILE is a structure (will be covered next week)
  - 

```
FILE *fp ;
fp = fopen("file1.txt", "r");
```

fopen()
open mode


"r"
파일의 시작
"w"
파일의 시작
(이미 있던 파일은 없어집니다.)
"a"
파일의 끝
(파일이 없을 경우는 새로 만듭니다.)

fopen()
open mode

SEOUL NATIONAL UNIVERSITY

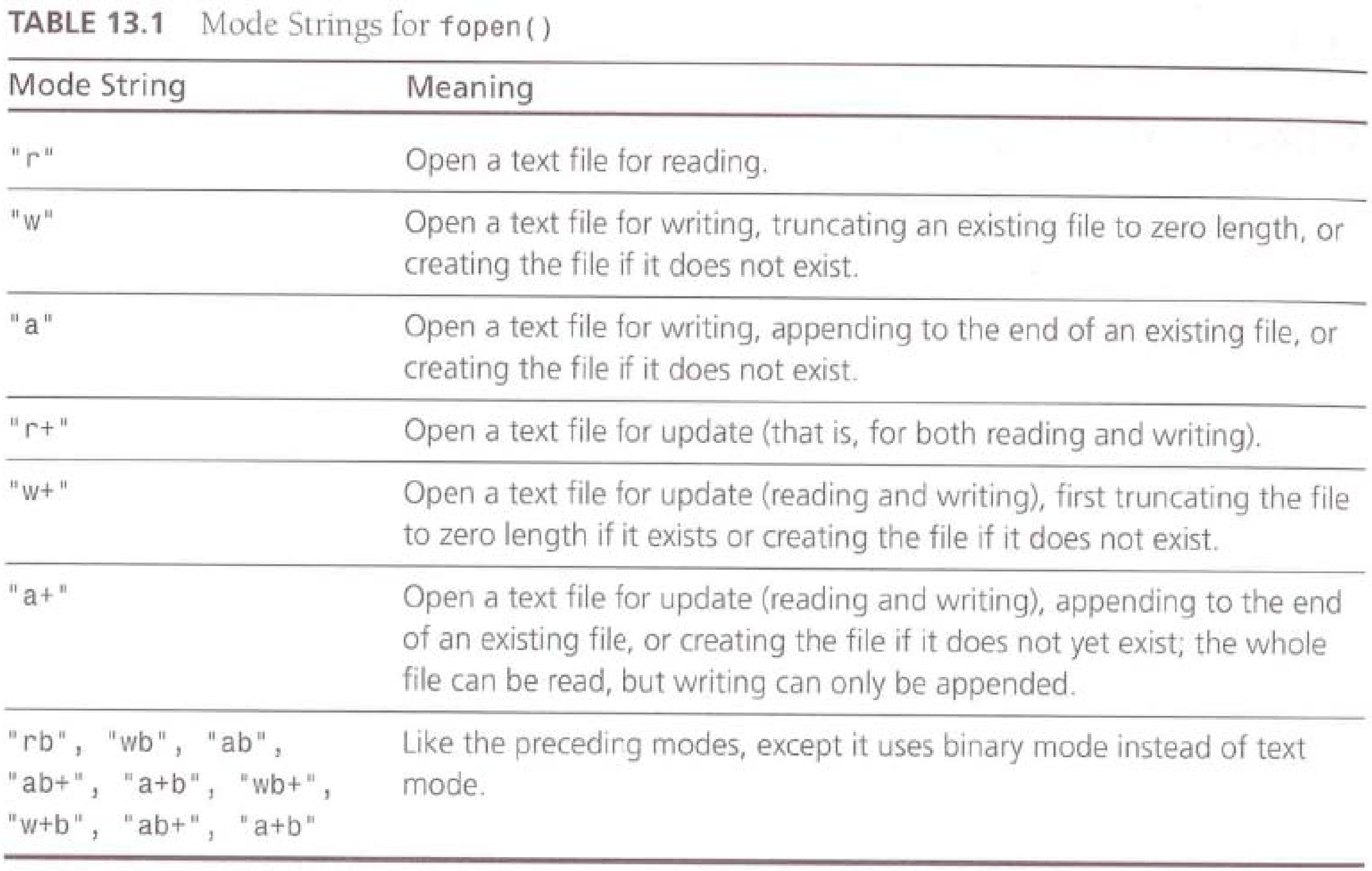


TABLE 13.1 Mode Strings for `fopen()`

Mode String	Meaning
`"r"`	Open a text file for reading.
`"w"`	Open a text file for writing, truncating an existing file to zero length, or creating the file if it does not exist.
`"a"`	Open a text file for writing, appending to the end of an existing file, or creating the file if it does not exist.
`"r+"`	Open a text file for update (that is, for both reading and writing).
`"w+"`	Open a text file for update (reading and writing), first truncating the file to zero length if it exists or creating the file if it does not exist.
`"a+"`	Open a text file for update (reading and writing), appending to the end of an existing file, or creating the file if it does not yet exist; the whole file can be read, but writing can only be appended.
`"rb"`, `"wb"`, `"ab"`, `"ab+"`, `"a+b"`, `"wb+"`, `"w+b"`, `"ab+"`, `"a+b"`	Like the preceding modes, except it uses binary mode instead of text mode.

fclose()

SEOUL NATIONAL UNIVERSITY

- fclose()
  - Closes the file
  - flushes buffers as needed
  - Returns a value of 0 if successful, and EOF if not.
  - Ex)

```
if (fclose(fp) != 0)
Printf(“%s 파일을 닫는데 에러가 발생했습니다.\n”, argv[1]);
```

  - Fclose() can file if
    - The disk is full, the floppy disk has been removed or there has been an I/O error

Other functions
fgets(), feof()

SEOUL NATIONAL UNIVERSITY

- fgets()
  - Reads input through the first new line character until one fewer than the upper limit of character is read, or until the end of file is found. (한줄씩 읽는다)
  - fgets(buf, MAX, fp);   ex) fgets(s,256,fp)

Name of char array

Pointer to FILE

Maximum size of the string

- feof()
  - Returns nonzero if the last input detected the end-of-file (true).
  - Returns zero otherwise (false)   ex) feof(fp)

Other functions
getc(), putc()

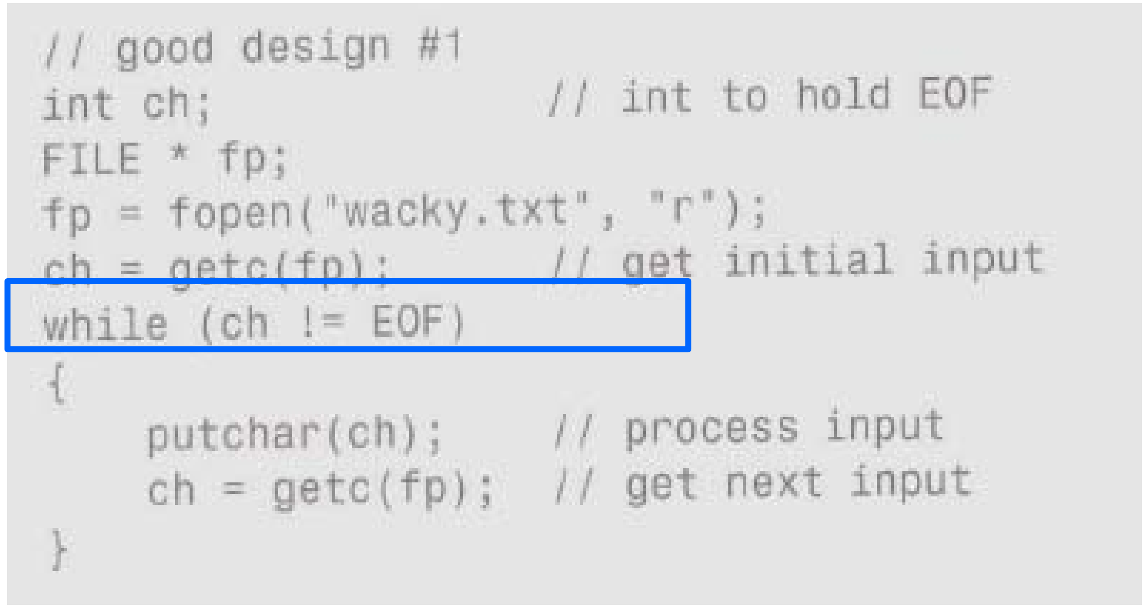
SEOUL NATIONAL UNIVERSITY

- getc(): get a character from a file
  - ch = getc(fp);   //need to tell which file you are using
  - ch = getchar();
- putc(): put the character into the file
  - putc(ch, fp);
  - putchar(ch);   =   putc(ch,stdout);

End-Of-File (EOF)

SEOUL NATIONAL UNIVERSITY

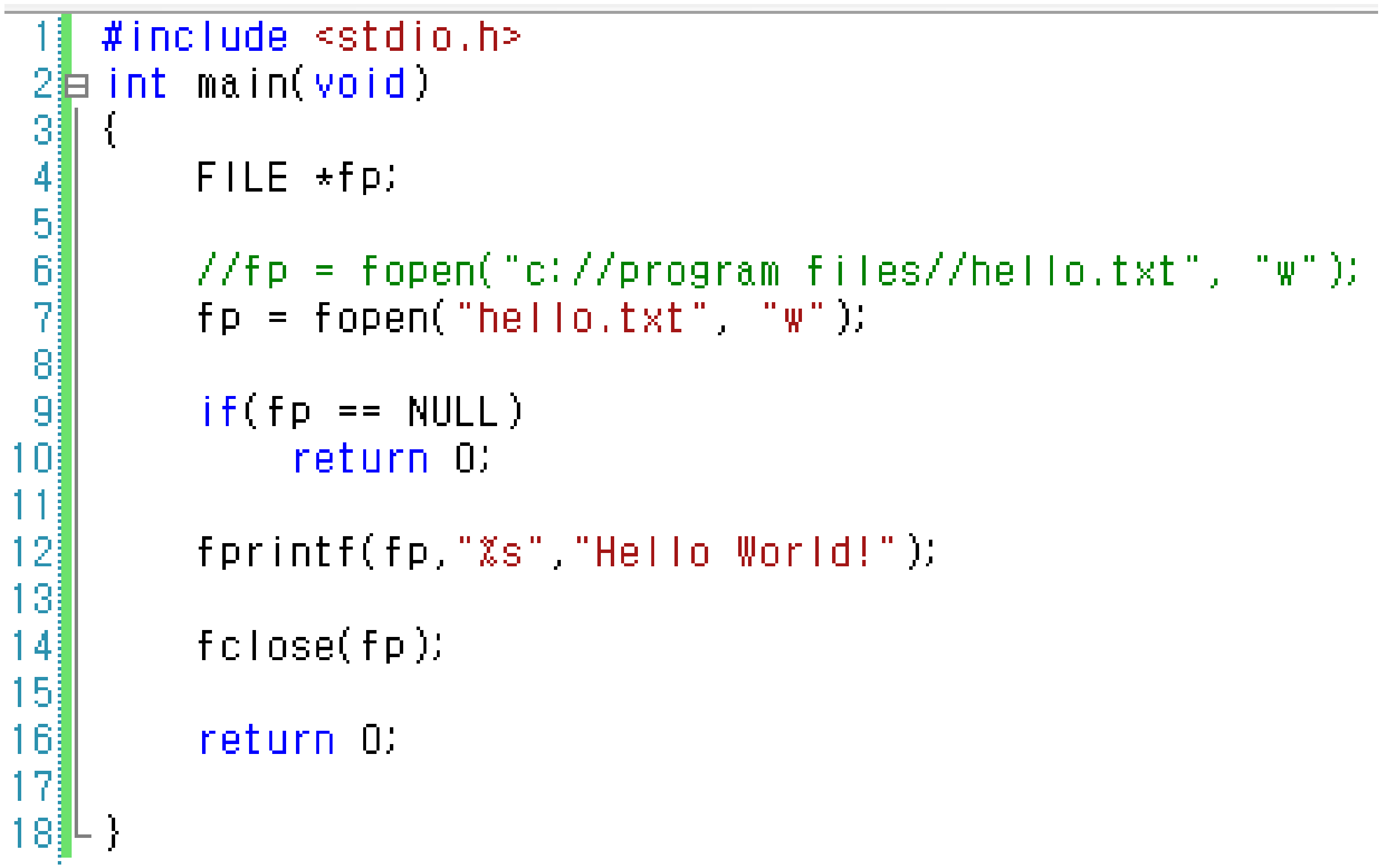
- The getc() function returns the special value EOF when it reach the end of the file
- To avoid problems attempting to read an empty file, use an entry-condition loop for file input.



```
// good design #1
int ch;                  // int to hold EOF
FILE * fp;
fp = fopen("wacky.txt", "r");
ch = getc(fp);           // get initial input
while (ch != EOF)
{
    putchar(ch);         // process input
    ch = getc(fp);       // get next input
}
```

Writing a file
an example (helloworld.c)

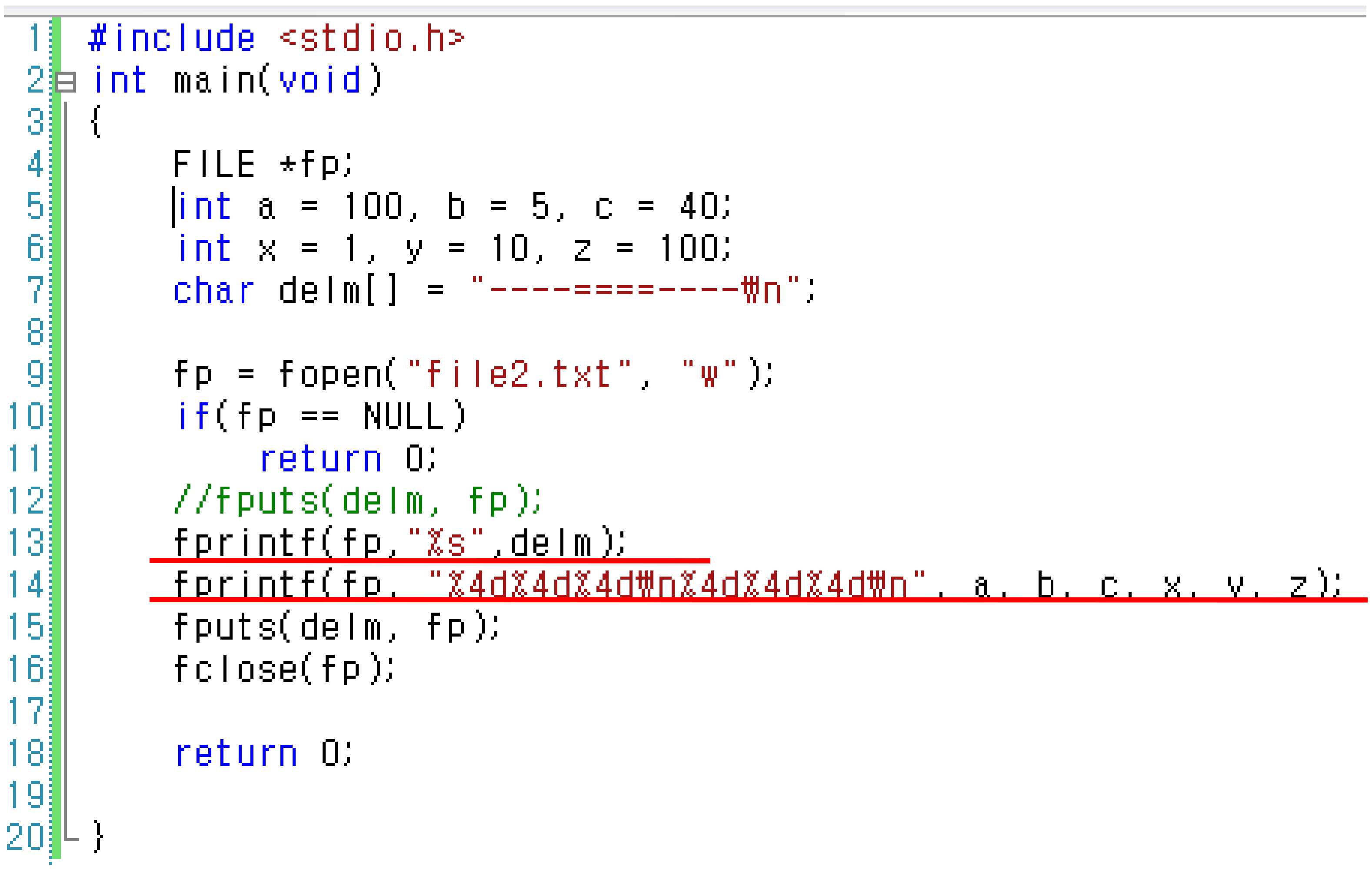
SEOUL NATIONAL UNIVERSITY



```c
#include <stdio.h>
int main(void)
{
	FILE *fp;

	//fp = fopen("c://program files//hello.txt", "w");
	fp = fopen("hello.txt", "w");

	if(fp == NULL)
		return 0;

	fprintf(fp,"%s","Hello World!");

	fclose(fp);

	return 0;

}
```

Writing a file

another example (write.c)



```c
#include <stdio.h>
int main(void)
{
	FILE *fp;
	int a = 100, b = 5, c = 40;
	int x = 1, y = 10, z = 100;
	char delm[] = "----====----₩n";

	fp = fopen("file2.txt", "w");
	if(fp == NULL)
		return 0;
	//fputs(delm, fp);
	fprintf(fp,"%s",delm);
	fprintf(fp, "%4d%4d%4d₩n%4d%4d%4d₩n", a, b, c, x, y, z);
	fputs(delm, fp);
	fclose(fp);

	return 0;

}
```

Reading/Writing a binary File

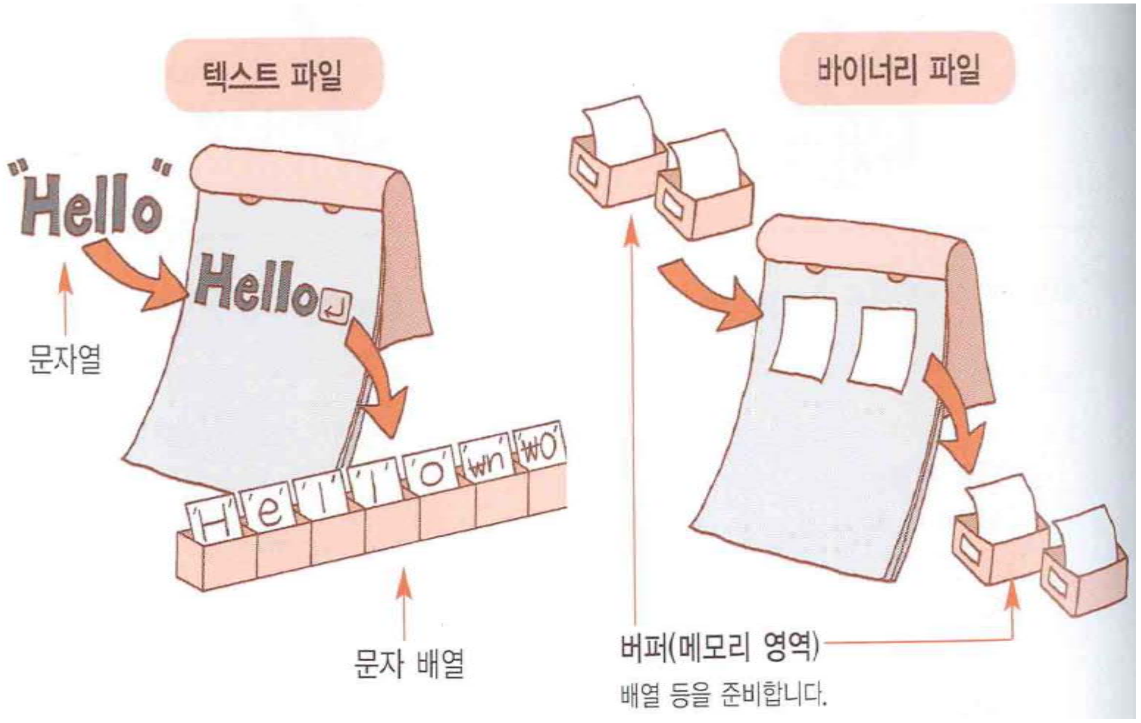
an example

SEOUL NATIONAL UNIVERSITY

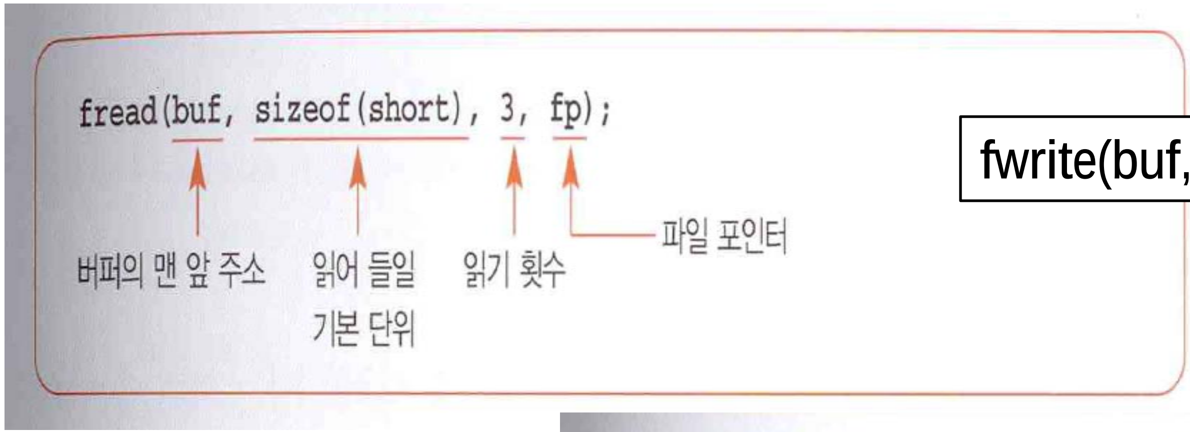
```
Double num = 1./3.;
fprintf(fp, “%f”, num);
```

- Saves num as a string of eight characters: 0.333333. or 0.33 with %.2f specifier → results in different values being stored.
- When a data is stored in a file using the sme representation that the program uses → data is stored in binary form
- 바이너리 파일은 문자, 행바꿈 문자, 제어 코드 등을 구별하지 않고 똑 같은 데이터로 취급한다.
- Add ‘b’ in open mode. Ex) fp = fopen(“file4.dat”, “wb”)

Reading/Writing a binary File
an example



Reading/Writing a binary File
an example



```
fread(buf, sizeof(short), 3, fp);
```

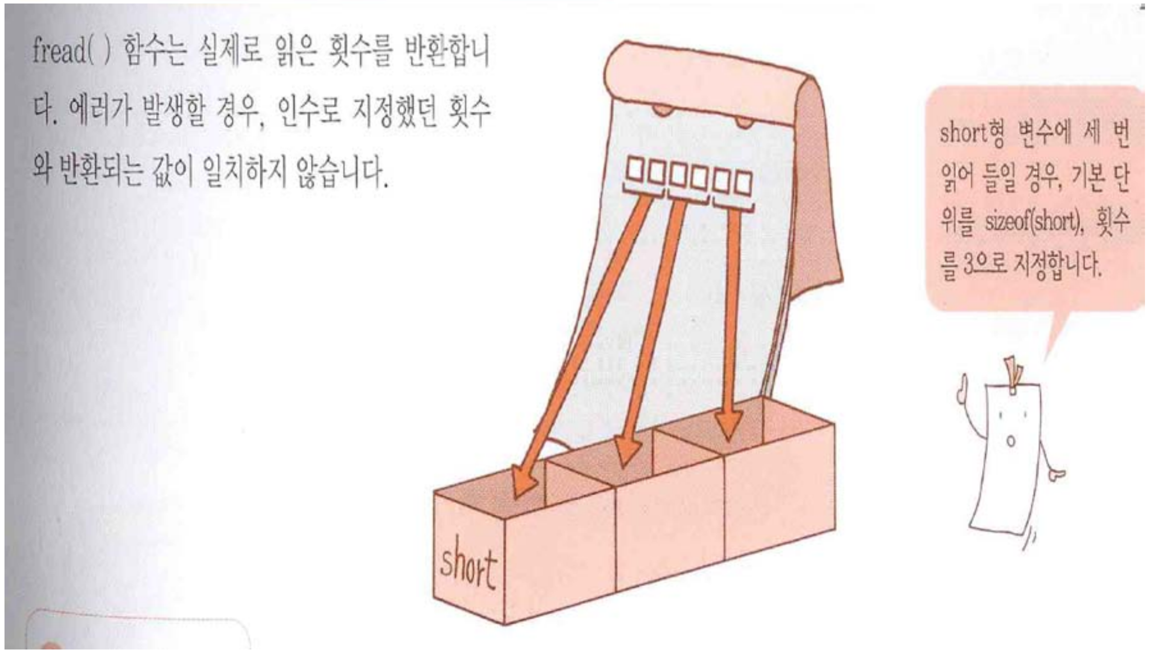
버퍼의 맨 앞 주소

읽어 들일 기본 단위

읽기 횟수

파일 포인터

```
fwrite(buf, sizeof(short), 3, fp)
```

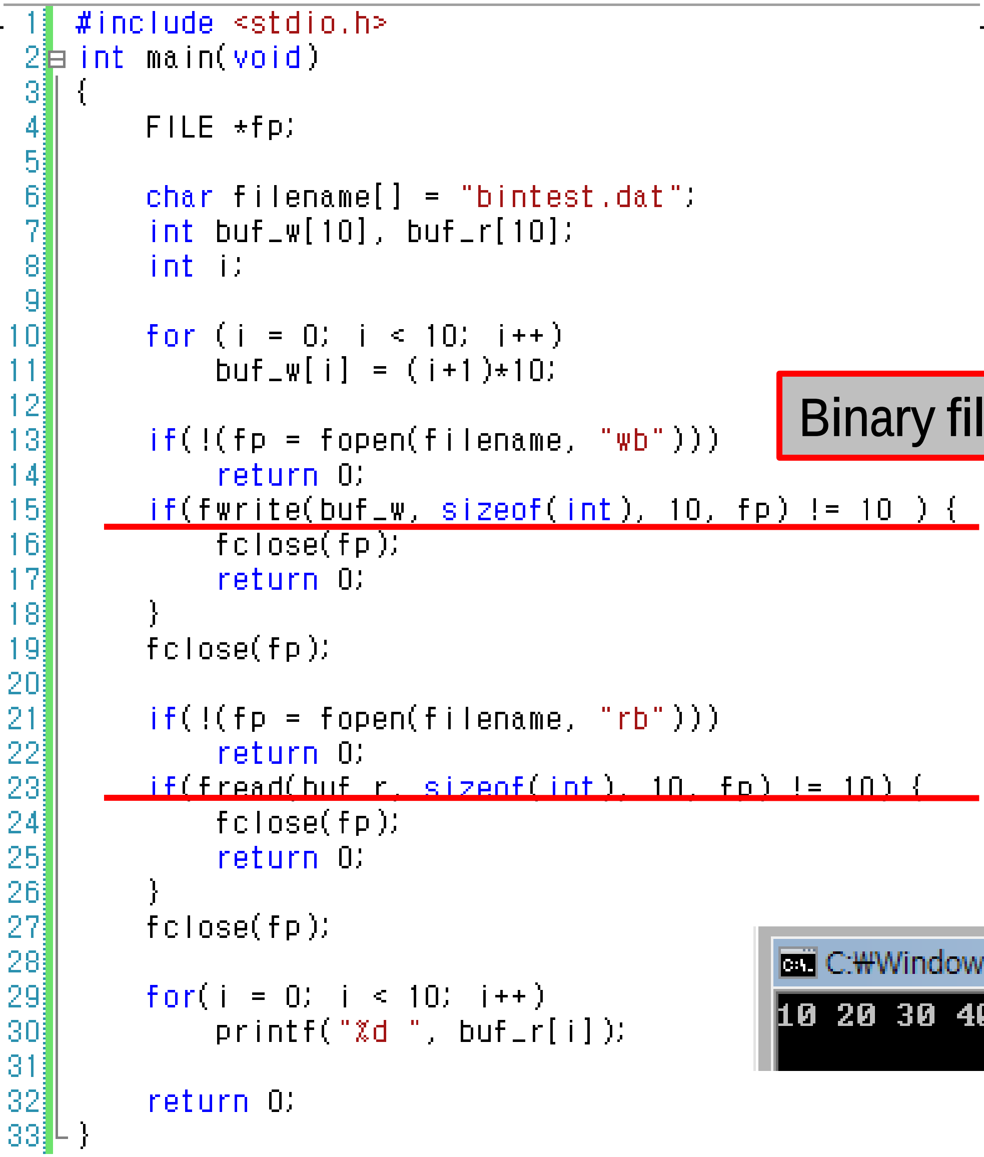


fread( ) 함수는 실제로 읽은 횟수를 반환합니다. 에러가 발생할 경우, 인수로 지정했던 횟수와 반환되는 값이 일치하지 않습니다.

short형 변수에 세 번 읽어 들일 경우, 기본 단위를 sizeof(short), 횟수를 3으로 지정합니다.

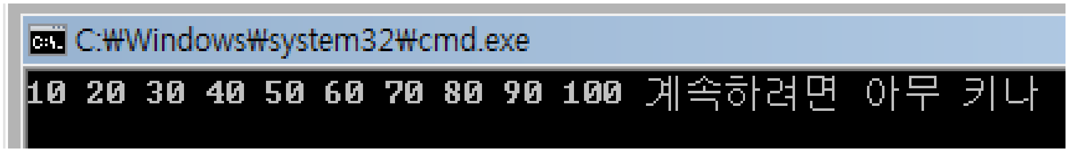
Reading/Writing a binary File
an example (binary.c)

SEOUL NATIONAL UNIVERSITY



```c
#include <stdio.h>
int main(void)
{
    FILE *fp;

    char filename[] = "bintest.dat";
    int buf_w[10], buf_r[10];
    int i;

    for (i = 0; i < 10; i++)
        buf_w[i] = (i+1)*10;

    if(!(fp = fopen(filename, "wb")))
        return 0;
    if(fwrite(buf_w, sizeof(int), 10, fp) != 10 ) {
        fclose(fp);
        return 0;
    }
    fclose(fp);

    if(!(fp = fopen(filename, "rb")))
        return 0;
    if(fread(buf_r, sizeof(int), 10, fp) != 10) {
        fclose(fp);
        return 0;
    }
    fclose(fp);

    for(i = 0; i < 10; i++)
        printf("%d ", buf_r[i]);

    return 0;
}
```

Binary file 에서는 fwrite()와 fread()를 쓴다.

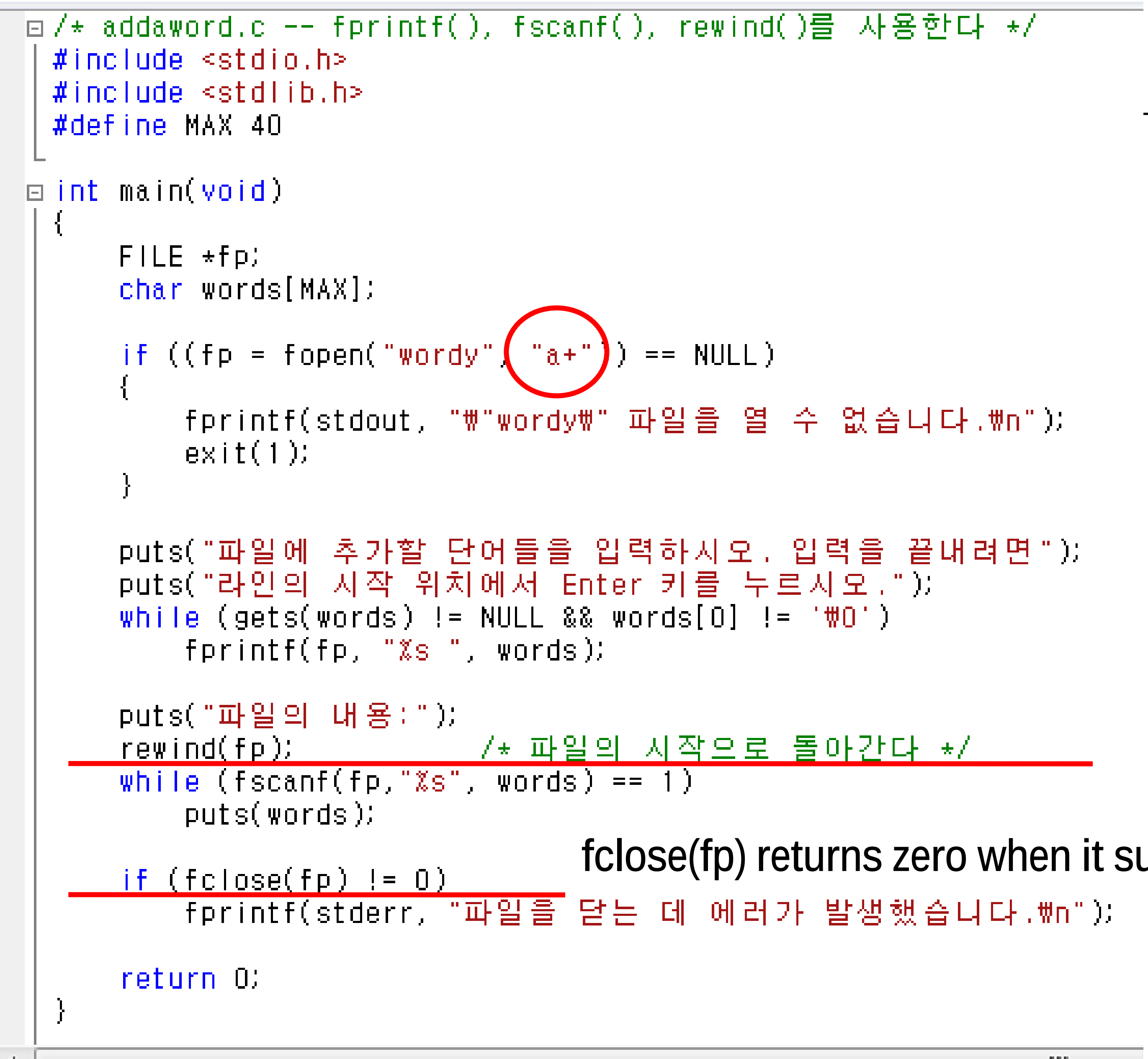


C:₩Windows₩system32₩cmd.exe

```
10 20 30 40 50 60 70 80 90 100 계속하려면 아무 키나
```

Appending

SEOUL NATIONAL UNIVERSITY

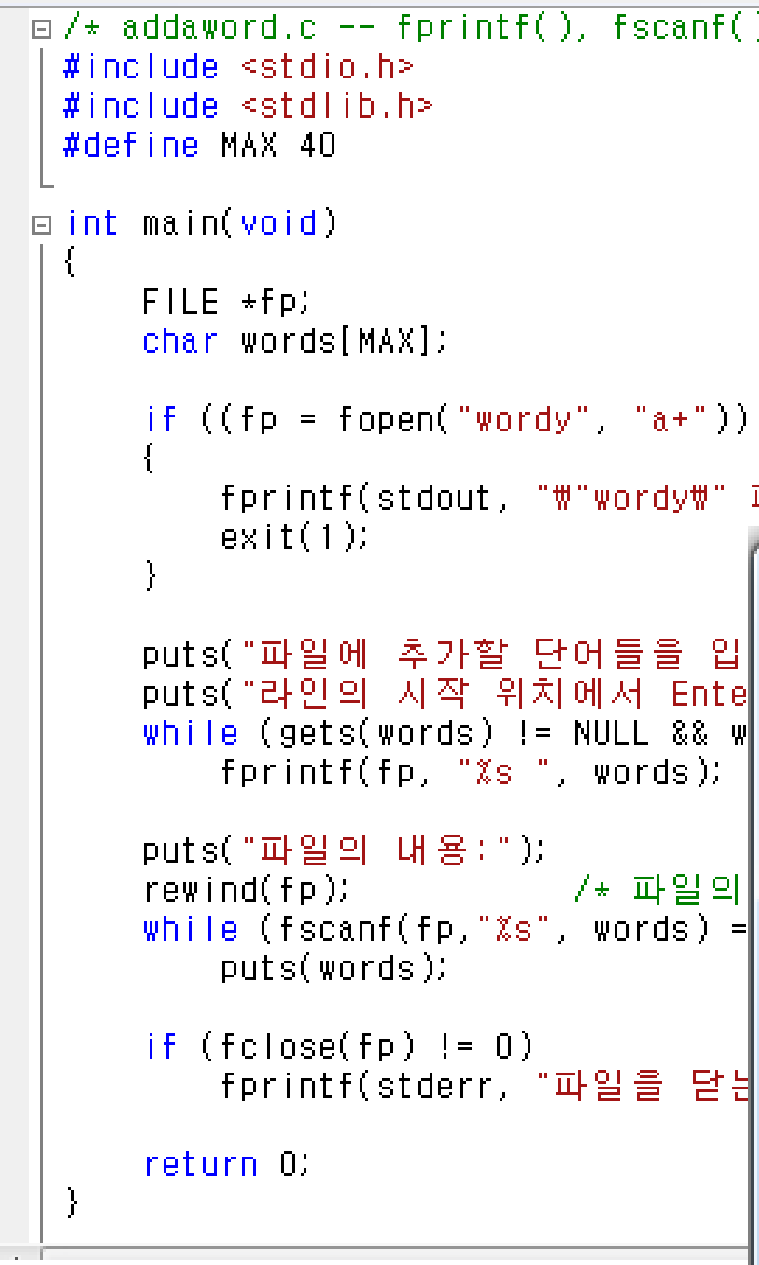


```c
/* addaword.c -- fprintf(), fscanf(), rewind()를 사용한다 */
#include <stdio.h>
#include <stdlib.h>
#define MAX 40

int main(void)
{
    FILE *fp;
    char words[MAX];

    if ((fp = fopen("wordy", "a+")) == NULL)
    {
        fprintf(stdout, "₩"wordy₩" 파일을 열 수 없습니다.₩n");
        exit(1);
    }

    puts("파일에 추가할 단어들을 입력하시오. 입력을 끝내려면");
    puts("라인의 시작 위치에서 Enter 키를 누르시오.");
    while (gets(words) != NULL && words[0] != '₩0')
        fprintf(fp, "%s ", words);

    puts("파일의 내용:");
    rewind(fp);             /* 파일의 시작으로 돌아간다 */
    while (fscanf(fp,"%s", words) == 1)
        puts(words);

    if (fclose(fp) != 0)
        fprintf(stderr, "파일을 닫는 데 에러가 발생했습니다.₩n");

    return 0;
}
```

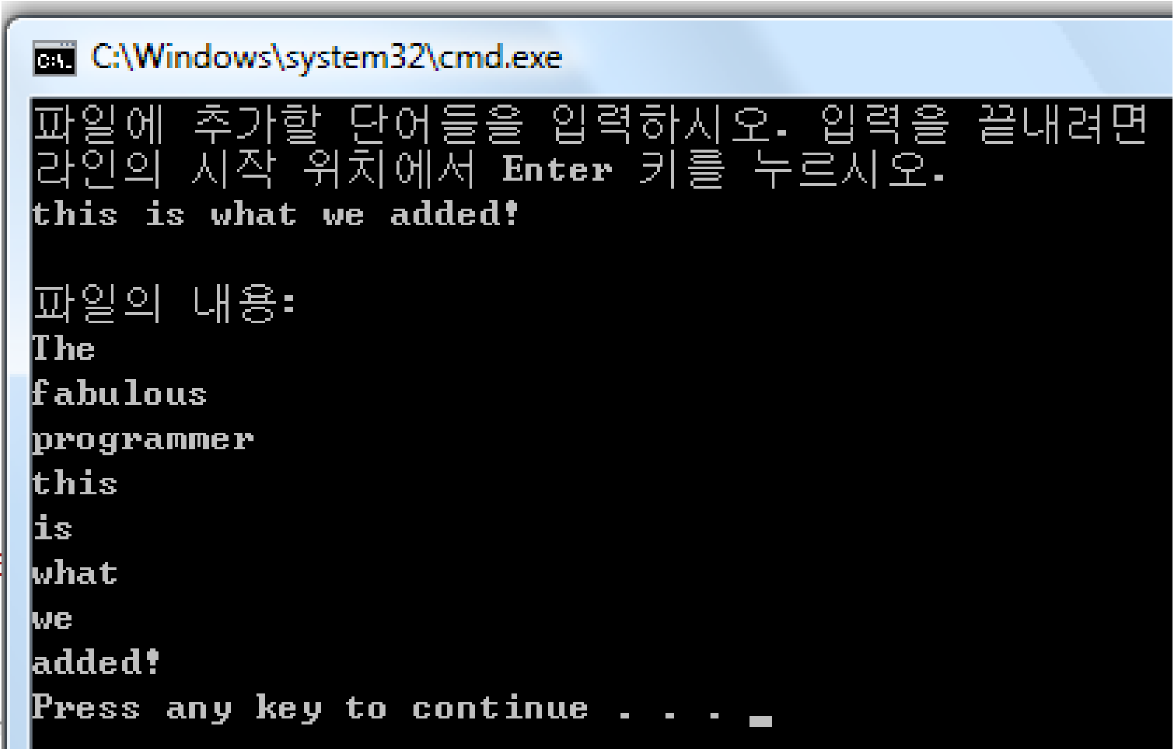
fclose(fp) returns zero when it succeed

Appending

SEOUL NATIONAL UNIVERSITY



```c
/* addaword.c -- fprintf(), fscanf(), rewind()를 사용한다 */
#include <stdio.h>
#include <stdlib.h>
#define MAX 40

int main(void)
{
    FILE *fp;
    char words[MAX];

    if ((fp = fopen("wordy", "a+")) == NULL)
    {
        fprintf(stdout, "₩"wordy₩" 파일을 열 수 없습니다.₩n");
        exit(1);
    }

    puts("파일에 추가할 단어들을 입
    puts("라인의 시작 위치에서 Ente
    while (gets(words) != NULL && w
        fprintf(fp, "%s ", words);

    puts("파일의 내용:");
    rewind(fp);              /* 파일의
    while (fscanf(fp,"%s", words) =
        puts(words);

    if (fclose(fp) != 0)
        fprintf(stderr, "파일을 닫

    return 0;
}
```

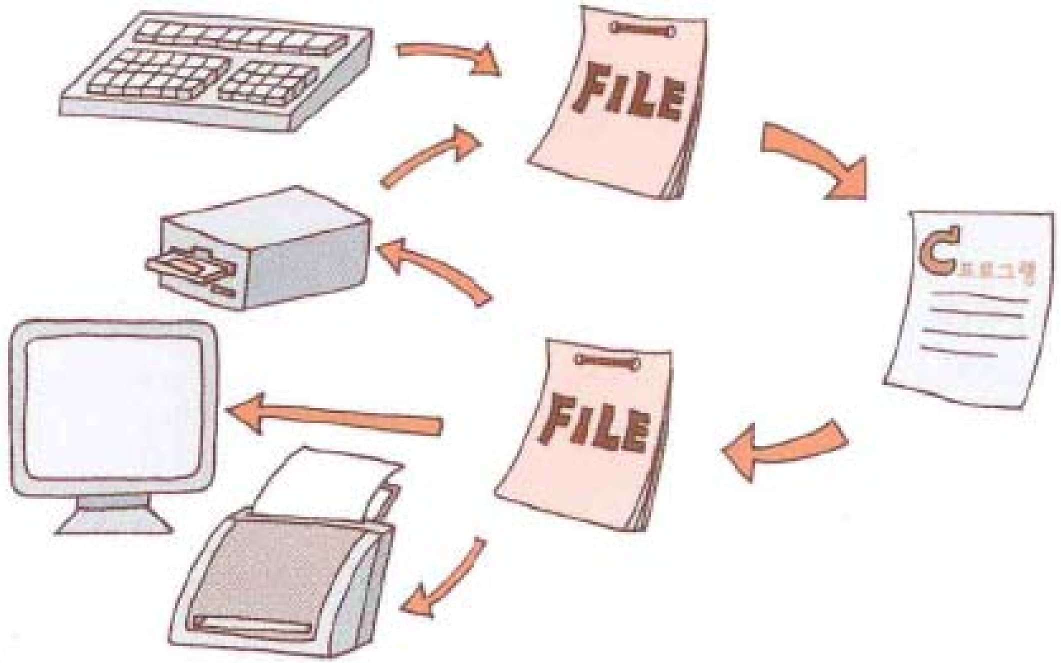


C:\Windows\system32\cmd.exe

```
파일에 추가할 단어들을 입력하시오. 입력을 끝내려면
라인의 시작 위치에서 Enter 키를 누르시오.
this is what we added!

파일의 내용:
The
fabulous
programmer
this
is
what
we
added!
Press any key to continue . . .
```

File
Standard Files

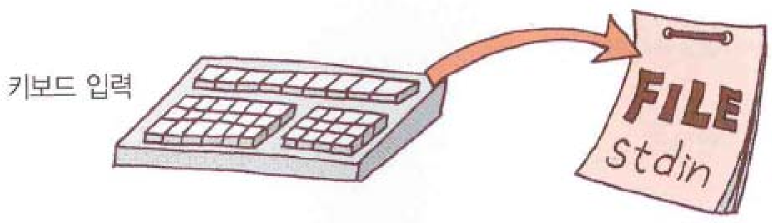
- In C, printer, monitor, keyboards can be considered as a ‘file’ --???
- There are three file pointers: stdin, stdout, stderr



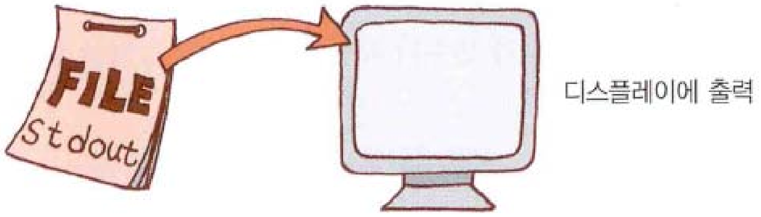
File
Standard Files

SEOUL NATIONAL UNIVERSITY

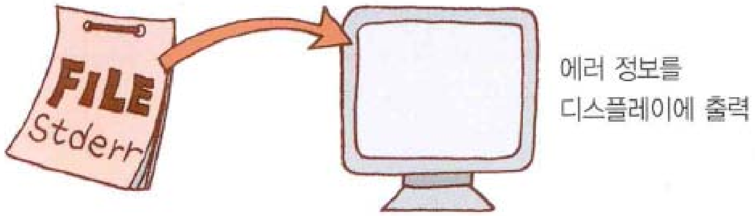
- Standard input
  - stdin: 키보드로부터 입력을 받는 파일 포인터



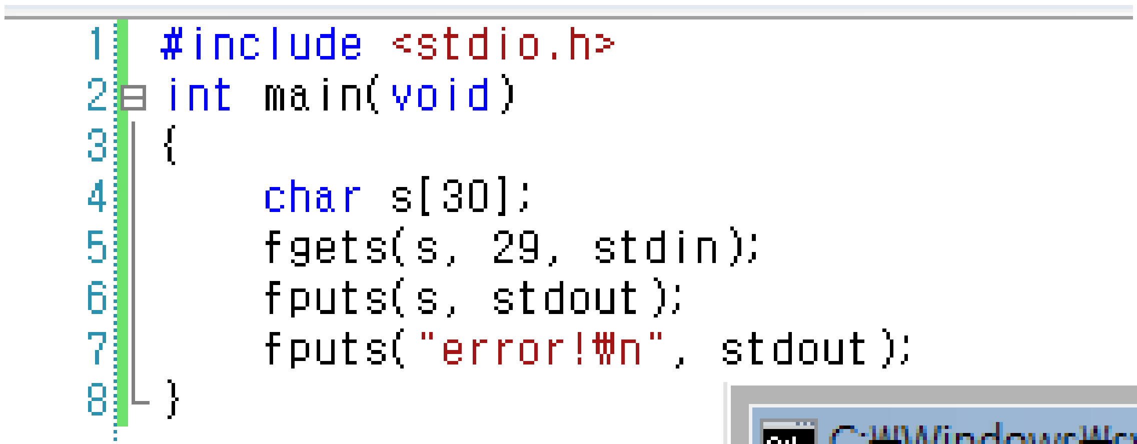
- Standard output file
  - stdout: 표준출력장치(모니터)에 출력할 때의 파일 포인터



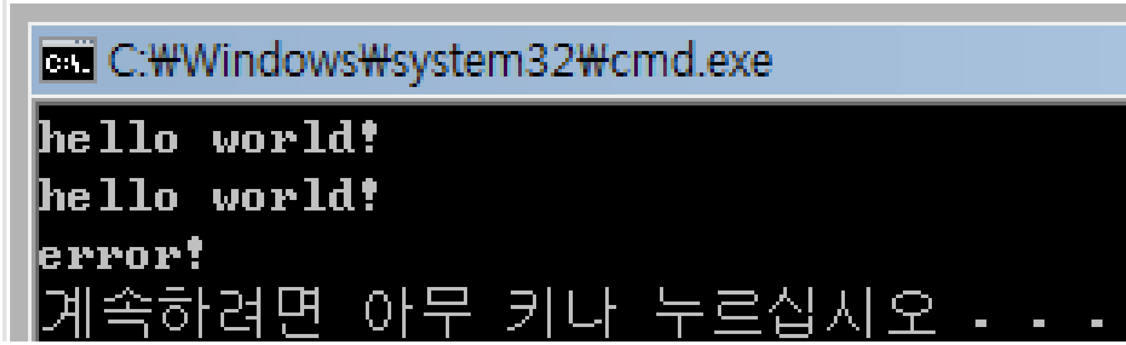
- Standard error output file
  - stderr: 표준에러추력장치(모니터)에 출력할 때의 파일 포인터



File
Standard Files



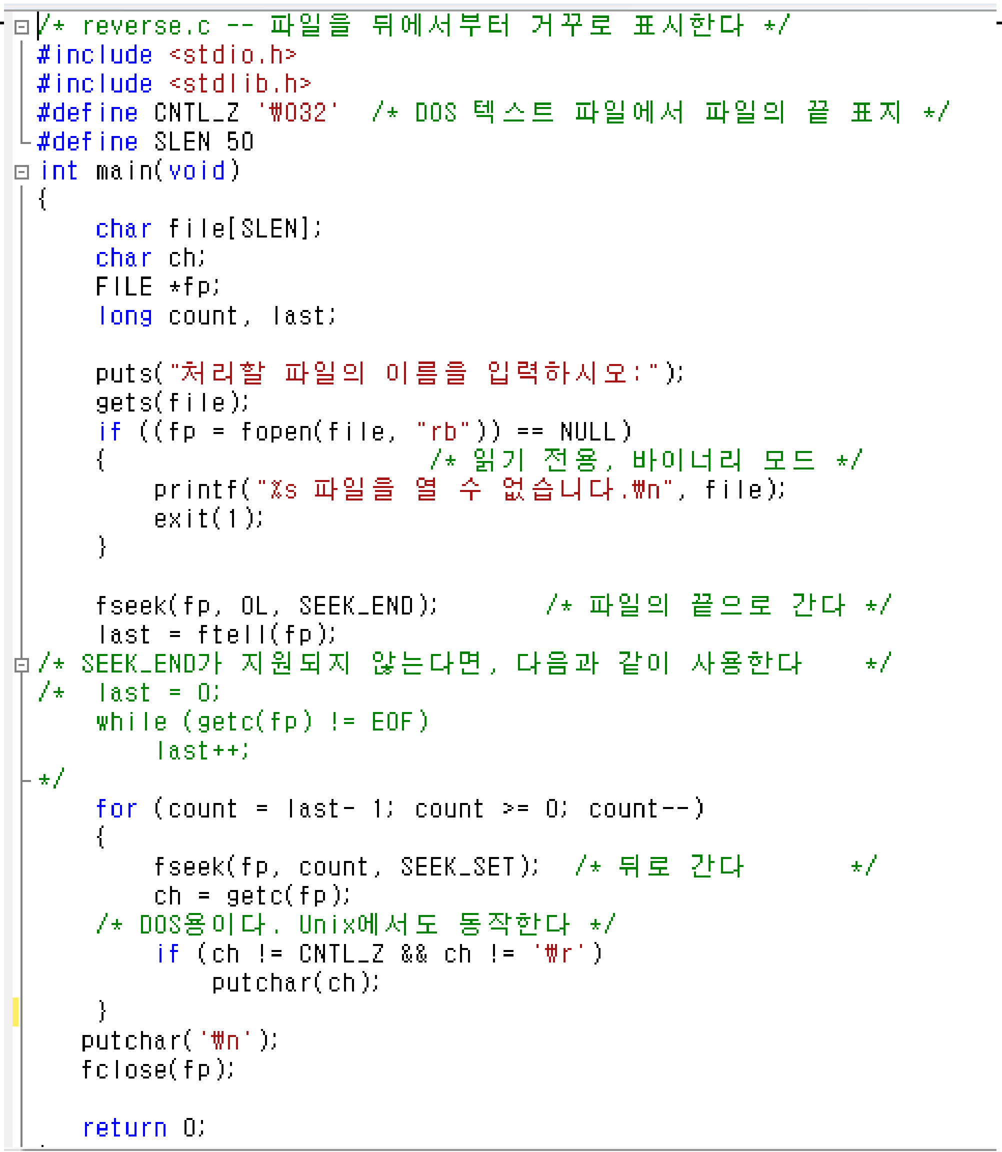
```c
#include <stdio.h>
int main(void)
{
    char s[30];
    fgets(s, 29, stdin);
    fputs(s, stdout);
    fputs("error!₩n", stdout);
}
```



```
hello world!
hello world!
error!
계속하려면 아무 키나 누르십시오 . . .
```

printf(“%s”,delm); ←→ fprintf(stdout, “%d”, delm)

Random access
fseek(), ftell() – reverse.c



```c
/* reverse.c -- 파일을 뒤에서부터 거꾸로 표시한다 */
#include <stdio.h>
#include <stdlib.h>
#define CNTL_Z '₩032'  /* DOS 텍스트 파일에서 파일의 끝 표지 */
#define SLEN 50
int main(void)
{
    char file[SLEN];
    char ch;
    FILE *fp;
    long count, last;

    puts("처리할 파일의 이름을 입력하시오:");
    gets(file);
    if ((fp = fopen(file, "rb")) == NULL)
    {                       /* 읽기 전용, 바이너리 모드 */
        printf("%s 파일을 열 수 없습니다.₩n", file);
        exit(1);
    }

    fseek(fp, 0L, SEEK_END);        /* 파일의 끝으로 간다 */
    last = ftell(fp);
/* SEEK_END가 지원되지 않는다면, 다음과 같이 사용한다     */
/*  last = 0;
    while (getc(fp) != EOF)
        last++;
*/
    for (count = last- 1; count >= 0; count--)
    {
        fseek(fp, count, SEEK_SET);  /* 뒤로 간다        */
        ch = getc(fp);
    /* DOS용이다. Unix에서도 동작한다 */
        if (ch != CNTL_Z && ch != '₩r')
            putchar(ch);
    }
   putchar('₩n');
   fclose(fp);

   return 0;
```

Random access
fseek(), ftell()

SEOUL NATIONAL UNIVERSITY

- fseek(FILE pointer, offset, mode)   ex) fseek(fp, 2L, SEEK_END)

How far to move from the starting point, in *long* type

SEEK_SET	: Beginning of file
SEEK_CUR	: Current position
SEEK_END	: End of file

- Ex)
- fseek(fp, 0L, SEEK_SET);   // go to the beginning of the file
- fseek(fp, 10L, SEEK_SET):   // go 10 bytes into the file
- fseek(fp, -10L, SEEK_END);   // back up 10 bytes from the end of the file

Random access
fseek(), ftell()

SEOUL NATIONAL UNIVERSITY

- Ftell() returns the current file location.

Summary of functions for I/O

SEOUL NATIONAL UNIVERSITY

function	Definition	form	Return (success/fail)
fopen()	Open the indicated file	fopen(“FILENAME”, “mode”)	a file pointer /NULL pointer
fclose	Close the indicated file	fclose(FILE *)	0 / EOF
fgetc()	Gets the next character from the indicated input stream	fgetc(FILE *)	Character/EOF
fputc()	writes the next character from the indicated input stream	fputc(int, FILE *)	Printed character/EOF
fgets()	Gets the next line from the indicated input stream	fgets(char *s, int n, FILE *)	Address of the string/NULL pointer
fputs()	Writes the character string pointed to by the first arguments to theindicated stream	fputs(const char *s, FILE *)	Last character/EOF

Summary of functions for I/O

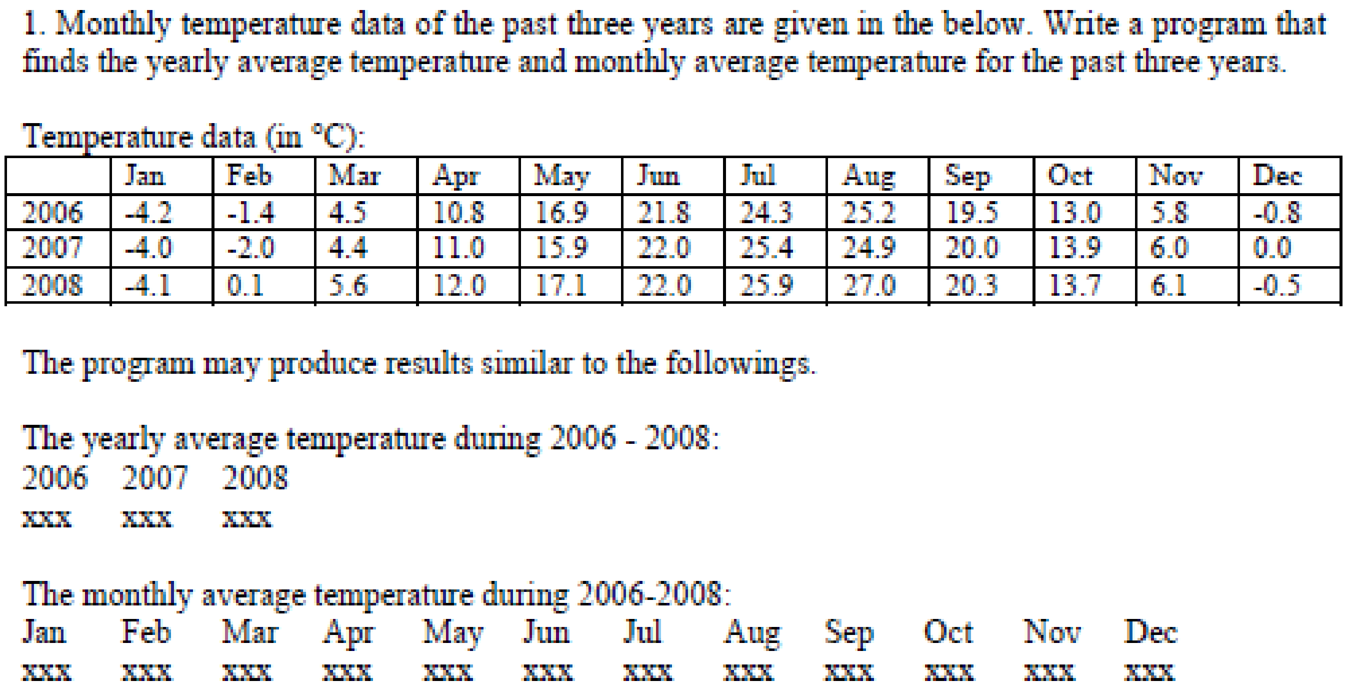
SEOUL NATIONAL UNIVERSITY

function	Definition	form	Return (success/fail)
gets()	gets the next line from the standard input	gets(char *s)	Address/NULL pointer
puts()	writes the string to the standard output	puts(char *s)	Non-negative value/ EOF
fprintf()	writes the formatted output to the indicated stream	fprintf(FILE *, format, argument)	Number of printed data/EOF
fscanf()	reads formatted input from the indicated stream	fscanf(FILE *, format, argument)	Number of scanned data/EOF
rewind()	sets the file-position pointer to the start of the file	rewind(FILE *)	X
fseek()	sets the file-position pointer to the indicated value	fseek(FILE *, offset, whence)	0/non-zero value
ftell()	gets the current file position	ftell(FILE *)	Bytes from the start/-1L

Exercise
Homework 6.

SEOUL NATIONAL UNIVERSITY

- Homework 6.1.
  - Rewrite your program for homework 5.1 so that the main tasks are performed by functions instead of in main(). Please refer to array2d.c in the textbook.



1. Monthly temperature data of the past three years are given in the below. Write a program that finds the yearly average temperature and monthly average temperature for the past three years.

Temperature data (in °C):

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2006	-4.2	-1.4	4.5	10.8	16.9	21.8	24.3	25.2	19.5	13.0	5.8	-0.8
2007	-4.0	-2.0	4.4	11.0	15.9	22.0	25.4	24.9	20.0	13.9	6.0	0.0
2008	-4.1	0.1	5.6	12.0	17.1	22.0	25.9	27.0	20.3	13.7	6.1	-0.5

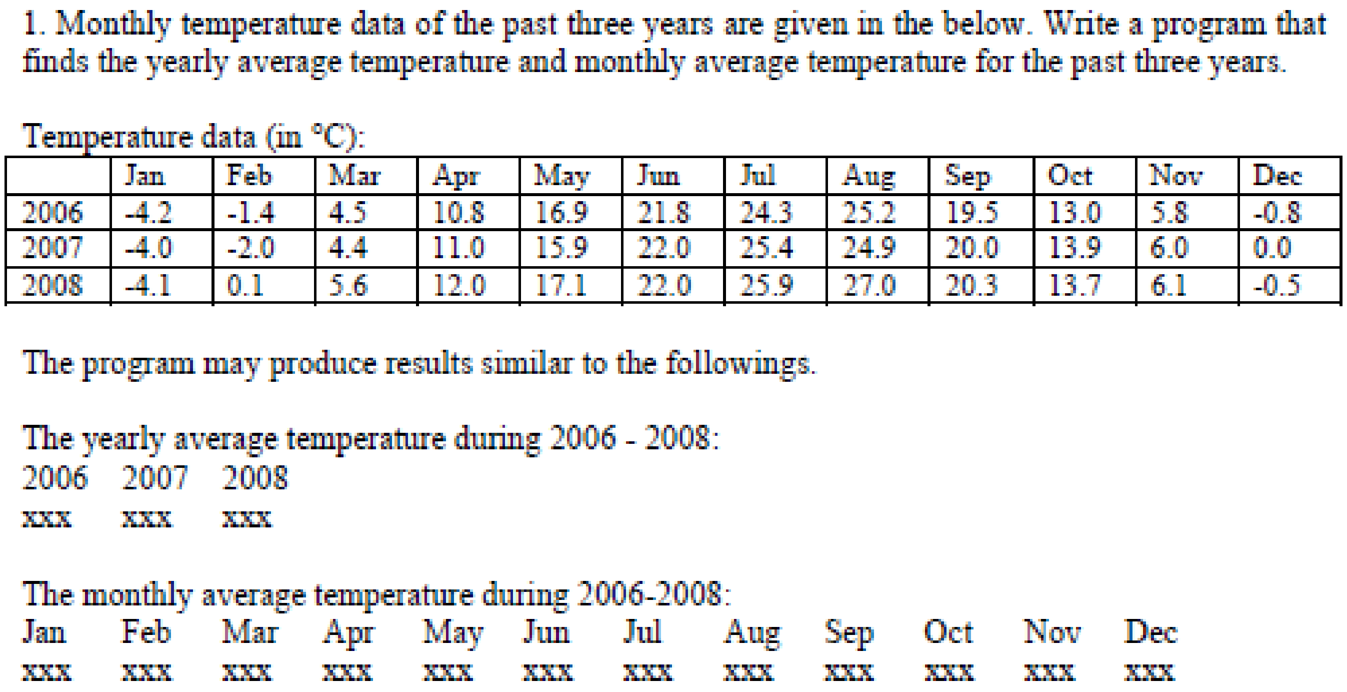
The program may produce results similar to the followings.

```
The yearly average temperature during 2006 - 2008:
2006    2007    2008
xxx     xxx     xxx

The monthly average temperature during 2006-2008:
Jan     Feb     Mar     Apr     May     Jun     Jul     Aug     Sep     Oct     Nov     Dec
xxx     xxx     xxx     xxx     xxx     xxx     xxx     xxx     xxx     xxx     xxx     xxx
```

Exercise Homework 6.

SEOUL NATIONAL UNIVERSITY

- Homework 6.2.
  - Rewrite your program for homework 5.1 using file input and output. Make a input text fie based on the temperature data and make a program that generate the output as a file format.



1. Monthly temperature data of the past three years are given in the below. Write a program that finds the yearly average temperature and monthly average temperature for the past three years.

Temperature data (in °C):

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2006	-4.2	-1.4	4.5	10.8	16.9	21.8	24.3	25.2	19.5	13.0	5.8	-0.8
2007	-4.0	-2.0	4.4	11.0	15.9	22.0	25.4	24.9	20.0	13.9	6.0	0.0
2008	-4.1	0.1	5.6	12.0	17.1	22.0	25.9	27.0	20.3	13.7	6.1	-0.5

The program may produce results similar to the followings.

```
The yearly average temperature during 2006 - 2008:
2006   2007   2008
xxx    xxx    xxx

The monthly average temperature during 2006-2008:
Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct    Nov    Dec
xxx    xxx    xxx    xxx    xxx    xxx    xxx    xxx    xxx    xxx    xxx    xxx
```

Today
Chapter 13

- File input and output
- Reading data from a file
- Writing data to a file
- Using Binary file
- Various functions: fopen(), fclose(), fgetc(), fgets(), fputc(), fputs(), gets(), puts(), fprintf(), fscanf(), rewind(), fseek(), ftell()