

컴퓨터 프로그래밍 및 실습

13주차. 스트림과 파일 입출력

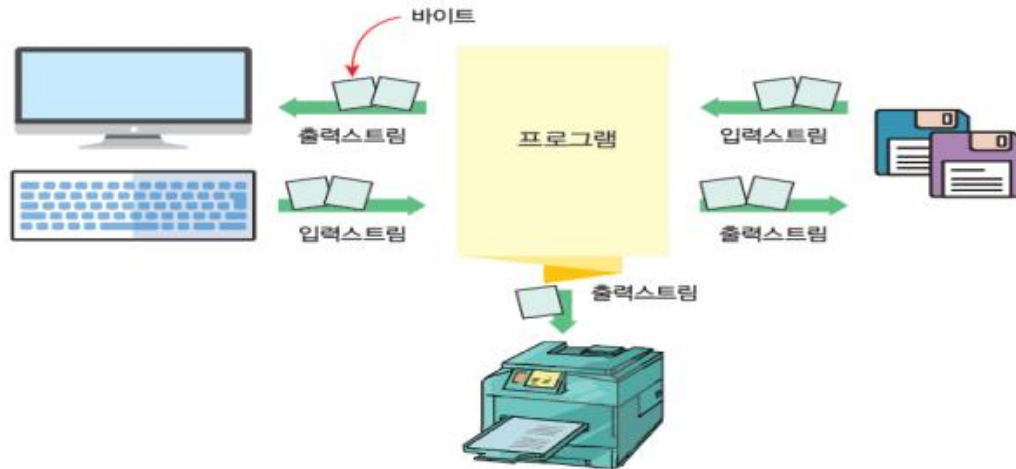
실습 안내

■ 실습 제출 안내

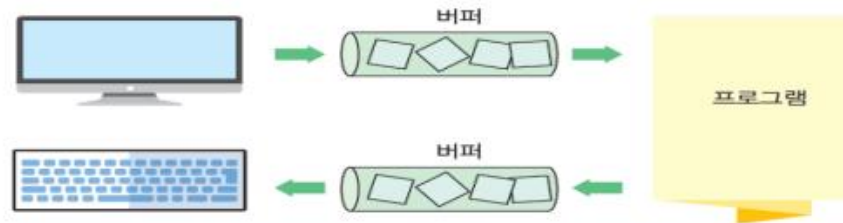
- 솔루션 이름은 "Practice week 13"
- 프로젝트 이름과 소스코드 이름은 Problem1, Problem2, ...
 - 실습1의 프로젝트 이름은 Problem1, 소스코드 이름은 problem1.c
 - 실습 2의 프로젝트 이름은 Problem2, 소스코드 이름은 problem2.c ...
- 솔루션 폴더를 압축하여 **Practice_week13_학번_이름.zip** 으로 제출
- 제출기한: 당일 **19시** 까지

1. 스트림 Stream

- ❖ 스트림(stream): 입력과 출력을 바이트(byte)들의 흐름으로 생각하는 것



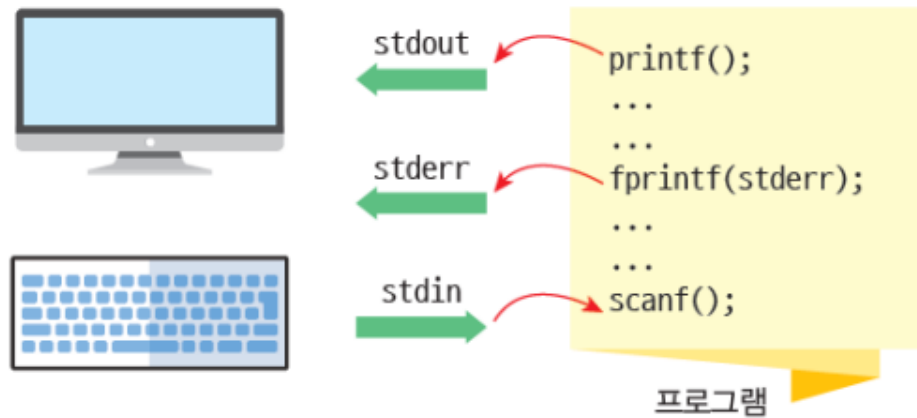
- ❖ 스트림에는 기본적으로 버퍼가 포함되어 있음



1. 스트림 Stream

❖ 표준 입출력 스트림(standard input/output stream)

이름	스트림	연결 장치
stdin	표준 입력 스트림	키보드
stdout	표준 출력 스트림	모니터의 화면
stderr	표준 오류 스트림	모니터의 화면



1. 스트림 Stream

❖ printf() 함수의 구문 `int printf(const char *format, ...);`

Syntax: printf()

예 `printf("%-10.3f", value);`

플래그 필드폭 정밀도 형식

❖ 플래그

기호	의미	기본값
-	출력 필드에서 출력값을 왼쪽 정렬한다.	오른쪽 정렬된다.
+	결과 값을 출력할 때 항상 +와 -의 부호를 붙인다.	음수일 때만 - 부호를 붙인다.
0	출력값 앞에 공백 문자 대신에 0으로 채운다. -와 0이 동시에 있으면 0은 무시된다. 만약 정수 출력의 경우, 정밀도가 지정되면 역시 0은 무시된다(예를 들어서 %08.5).	채우지 않는다.
blank(' ')	출력값 앞에 양수나 영인 경우에는 부호대신 공백을 출력한다. 음수일 때는 -가 붙여진다. + 플래그가 있으면 무시된다.	공백을 출력하지 않는다.
#	8진수 출력 시에는 출력값 앞에 0을 붙이고 16진수 출력 시에는 0x를 붙인다.	붙이지 않는다.

1. 스트림 Stream

❖ 형식 지정자

형식 지정자	설명	출력 예
%d	부호있는 10진수 형식으로 출력	255
%i	부호있는 10진수 형식으로 출력	255
%u	부호없는 10진수 형식으로 출력	255
%o	부호없는 8진수 형식으로 출력	377
%x	부호없는 16진수 형식으로 출력, 소문자로 표기	le
%X	부호없는 16진수 형식으로 출력, 대문자로 표기	FE
%f	소수점 고정 표기 형식으로 출력	123.456
%e	지수 표기 형식으로 출력, 지수 부분을 e로 표시	1.23456e+2
%E	지수 표기 형식으로 출력, 지수 부분을 E로 표시	1.23456E+2
%g	%e형식과 %f 형식 중 더 짧은 형식으로 출력	123.456
%G	%E형식과 %f 형식 중 더 짧은 형식으로 출력	123.456
%p	포인터 형식으로 출력	0027FDD0

1. 스트림 Stream

■ 실습 1

- 0~30 까지의 숫자를 아래 처럼 출력
 - 01
 - 02
 - ... 30
- 0~30 까지의 숫자를 아래 처럼 출력

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

1. 스트림 Stream

■ 실습 1

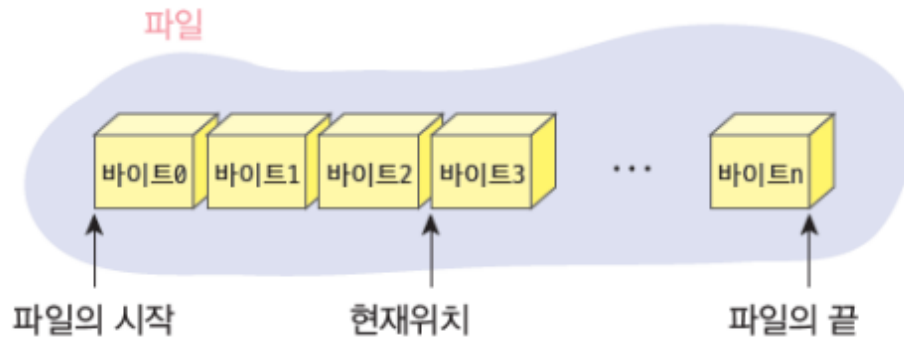
- 0.01~0.008 까지의 실수를 0.0002 단위로 아래처럼 출력

```
1.0e-02  
9.8e-03  
9.6e-03  
9.4e-03  
9.2e-03  
9.0e-03  
8.8e-03  
8.6e-03  
8.4e-03  
8.2e-03  
8.0e-03
```


2. 파일

❖ 파일의 개념

- C에서의 파일은 일련의 연속된 바이트
- 모든 파일 데이터들은 결국은 바이트로 바뀌어서 파일에 저장
- 이들 바이트들을 어떻게 해석하느냐는 전적으로 프로그래머의 책임



2. 파일

❖ 파일 처리의 개요

- 파일을 다룰 때는 반드시 다음과 같은 순서를 지켜야 함
 - ✓ 파일 열기 -> 파일 읽기와 쓰기 -> 파일 닫기
- 디스크 파일은 FILE 구조체를 이용하여 접근
- FILE 구조체를 가리키는 포인터를 파일 포인터(file pointer)

• 파일 열기

Syntax: 파일열기

예 `FILE *fp;
fp = fopen("test.txt", "w");`

파일 이름

파일 모드

2. 파일

❖ 파일 모드

모드	설명
"r"	읽기 모드로 파일을 연다. 만약 파일이 존재하지 않으면 오류가 발생한다.
"w"	쓰기 모드로 새로운 파일을 생성한다. 파일이 이미 존재하면 기존의 내용이 지워진다.
"a"	추가 모드로 파일을 연다. 만약 기존의 파일이 있으면 데이터가 파일의 끝에 추가된다. 파일이 없으면 새로운 파일을 만든다.
"r+"	읽기 모드로 파일을 연다. 쓰기 모드로 전환할 수 있다. 파일이 반드시 존재하여야 한다.
"w+"	쓰기 모드로 새로운 파일을 생성한다. 읽기 모드로 전환할 수 있다. 파일이 이미 존재하면 기존의 내용이 지워진다.
"a+"	추가 모드로 파일을 연다. 읽기 모드로 전환할 수 있다. 데이터를 추가하면 EOF 마커를 추가된 데이터의 뒤로 이동한다. 파일이 없으면 새로운 파일을 만든다.
"t"	텍스트 파일 모드로 파일을 연다.
"b"	이진 파일 모드로 파일을 연다.

2. 파일

❖ 파일 닫기

Syntax: 파일닫기

예

`fclose(fp):`

FILE 포인터

2. 파일

■ 파일 다루기

- 파일을 열 때는 fopen() 함수에 path와 mode 제공
- fopen() 호출 이후에도 fp가 NULL이라면
- 파일 열기 실패
- 파일을 다루고 난 이후에는 fclose 호출해야함

```
#include <stdio.h>

int main(void) {
    FILE* fp = NULL;

    fp = fopen("sample.txt", "r");

    if (fp == NULL) {
        printf("파일 열기 실패\n");
    } else {
        printf("파일 열기 성공\n");
    }

    // do something ...

    fclose(fp);
    return 0;
}
```

2. 파일

❖ 기타 유용한 함수들

함수	설명
<code>int foef(FILE *stream)</code>	파일의 끝이 도달되면 true를 반환한다.
<code>int rename(const char *oldname, const char *newname)</code>	파일의 이름을 변경한다.
<code>FILE *tmpfile()</code>	임시 파일을 생성하여 반환한다.
<code>int ferror(FILE *stream)</code>	스트림의 오류 상태를 반환한다. 오류가 발생하면 true가 반환된다.

2. 파일

❖ 기타 유용한 함수들

함수	설명
<code>int foef(FILE *stream)</code>	파일의 끝이 도달되면 true를 반환한다.
<code>int rename(const char *oldname, const char *newname)</code>	파일의 이름을 변경한다.
<code>FILE *tmpfile()</code>	임시 파일을 생성하여 반환한다.
<code>int ferror(FILE *stream)</code>	스트림의 오류 상태를 반환한다. 오류가 발생하면 true가 반환된다.

2. 파일

❖ 파일 입출력 함수

종류	입력 함수	출력 함수
문자 단위	<code>int fgetc(FILE *fp)</code>	<code>int fputc(int c, FILE *fp)</code>
문자열 단위	<code>char *fgets(char *buf, int n, FILE *fp)</code>	<code>int fputs(const char *buf, FILE *fp)</code>
서식화된 입출력	<code>int fscanf(FILE *fp, ...)</code>	<code>int fprintf(FILE *fp,...)</code>
이진 데이터	<code>size_t fread(char *buffer, int size, int count, FILE *fp)</code>	<code>size_t fwrite(char *buffer, int size, int count, FILE *fp)</code>

2. 파일

- fopen() 함수 사용 시에는 #define _CRT_SECURE_NO_WARNINGS 정의 필수

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

int main(void) {
    FILE* fp = fopen("test.txt", "w");

    if (fp == NULL) {
        printf("파일 열기 실패\n");
        return 0;
    }

    fputs("Hello World\n", fp);

    for (int i = 0; i < 10; i++) {
        fprintf(fp, "%d\n", i);
    }

    fclose(fp);

    return 0;
}
```

test.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보:

Hello World

0
1
2
3
4
5
6
7
8
9

2. 파일

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

int main(void) {
    FILE* fp = fopen("./test.txt", "r");

    if (fp == NULL) {
        printf("error\n");
        return 0;
    }

    while (!feof(fp)) {
        int data;
        fscanf(fp, "%d", &data);
        printf("%d ", data);
    }

    return 0;
}
```

test.txt - Windows [

파일(F) 편집(E) 서식

1
5
4
2
1
8
9
5
1
7
8

Microsoft Visual Studio 디버그 콘솔

1 5 4 2 1 8 9 5 1 7 8
D:\git\WC-TWC Programming\Pr
(코드: 0개).

2. 파일

■ 실습 2

- Data.txt 에는 학생 이름, 학번, 점수가 담겨 있다.
- 이를 Qsort를 이용하여 점수 순으로 정렬 하고, 평균을 계산하여 Result.txt에 저장하는 프로그램을 개발해보자.

Data.txt - Windows 메모장

파일(F)	편집(E)	서식(O)
202201	김철수	80
202105	박민수	15
202203	이수민	91
201301	김갑수	50
202009	정민철	87

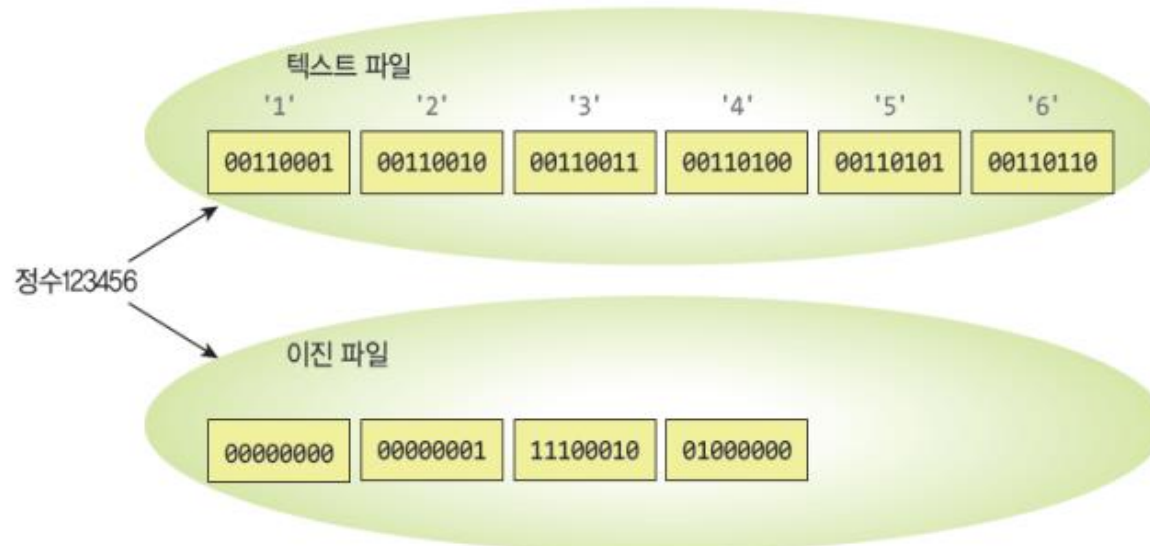
Result.txt - Windows 메모장

파일(F)	편집(E)	서식(O)	도움말(H)
202203	이수민	91	
202009	정민철	87	
202201	김철수	80	
201301	김갑수	50	
202105	박민수	15	
평균: 64.6			

3. 이진 파일

❖ 텍스트 파일과 이진 파일의 차이

- 텍스트 파일: 모든 데이터가 아스키 코드(문자열)로 변환되어서 저장됨
- 이진 파일: 컴퓨터에서 데이터를 표현하는 방식 그대로 저장



각 픽셀의 밝기를 2진수로 나타낸다.

3. 이진 파일

❖ 이진 파일 열기 (1/2)

- `fp = fopen("binary.bin", "wb");`
- 이진 파일 모드

파일 모드	설명
"rb"	읽기 모드 + 이진 파일 모드
"wb"	쓰기 모드 + 이진 파일 모드
"ab"	추가 모드 + 이진 파일 모드
"rb+"	읽고 쓰기 모드 + 이진 파일 모드
"wb+"	쓰고 읽기 모드 + 이진 파일 모드

3. 이진 파일

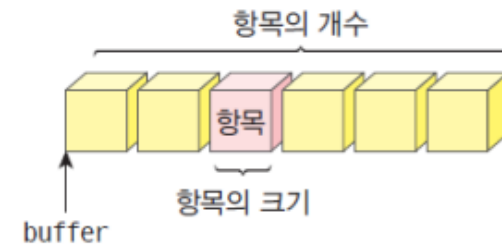
❖ 이진 파일 쓰기 (2/2)

Syntax: fwrite()

예 fwrite(buffer, sizeof(int), SIZE, fp);

메모리 블록의 주소 항목의 크기 항목의 개수 FILE 포인터

- buffer는 파일에 기록할 데이터를 가지고 있는 메모리 블록
- size는 저장되는 항목의 크기로서 단위는 바이트
- count는 읽으려고 하는 항목의 개수
- fp는 FILE 포인터



3. 이진 파일

❖ 이진 파일 읽기

Syntax: fread()

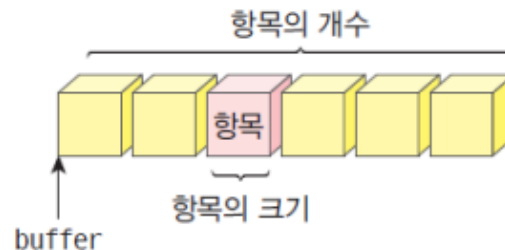
예 fread(buffer, sizeof(int), SIZE, fp);

메모리 블록의 주소

항목의 크기

항목의 개수

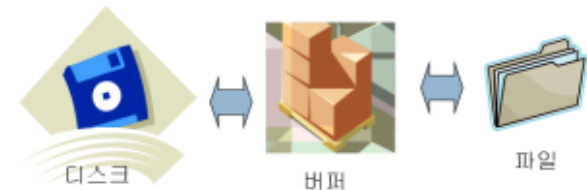
FILE 포인터



3. 이진 파일

❖ 버퍼링(buffering)

- 버퍼는 파일로부터 읽고 쓰는 데이터의 임시 저장 장소로 이용되는 메모리의 블록
- 디스크 드라이브는 블록 단위 장치이기 때문에 블록 단위로 입출력을 해야만 가장 효율적으로 동작
- 1024바이트의 블록이 일반적임
- `fflush(fp);`
 - ✓ 버퍼의 내용이 디스크 파일에 기록됨
- `Setbuf(fp, NULL);`
 - ✓ `Setbuf()`는 스트림의 버퍼를 직접 지정하는 함수로서 만약 버퍼 자리에 `NULL`을 써주면 버퍼를 제거하겠다는 것을 의미



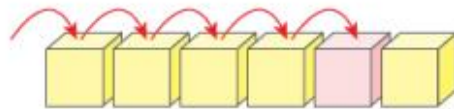
4. 파일 임의 접근

❖ 순차 접근(sequential access) 방법

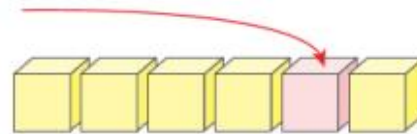
- 데이터를 파일의 처음부터 순차적으로 읽거나 기록하는 방법

❖ 임의 접근(random access) 방법

- 파일의 어느 위치에서든 읽기와 쓰기가 가능한 방법



순차 접근파일



임의 접근파일

4. 파일 임의 접근

■ 파일 커서

- 파일의 어디까지 읽었나?

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

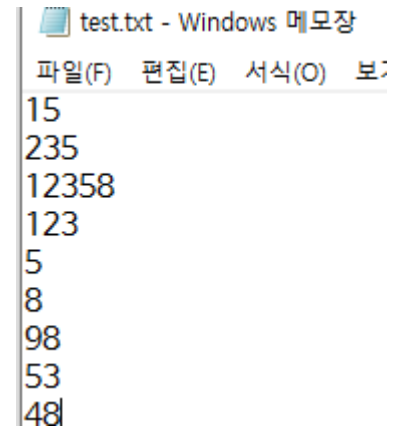
int main(void)
{
    FILE* fp = fopen("test.txt", "r");

    int temp;
    fscanf(fp, "%d", &temp);
    fscanf(fp, "%d", &temp);

    int cur = ftell(fp);    // 파일 커서의 현재 위치 반환

    printf("%d\n", cur);

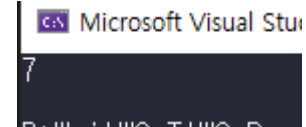
    return 0;
}
```



test.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보

15
235
12358
123
5
8
98
53
48



Microsoft Visual Studio

7

4. 파일 임의 접근

■ 파일 커서

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

int main(void)
{
    FILE* fp = fopen("test.txt", "r");

    int temp;
    fscanf(fp, "%d", &temp);
    fscanf(fp, "%d", &temp);

    int cur = ftell(fp);

    printf("%d\n", cur);

    rewind(fp); // 파일 커서를 파일의 처음 부분으로 돌림

    fscanf(fp, "%d", &temp);

    printf("%d\n", temp);

    return 0;
}
```

test.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보...

15
235
12358
123
5
8
98
53
48

Microsoft Visual Studi

7
15

D:\Program Files\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.29.30133\bin\Hostx64-x64\cmd.exe

4. 파일 임의 접근

■ 파일 커서

❖ fseek()

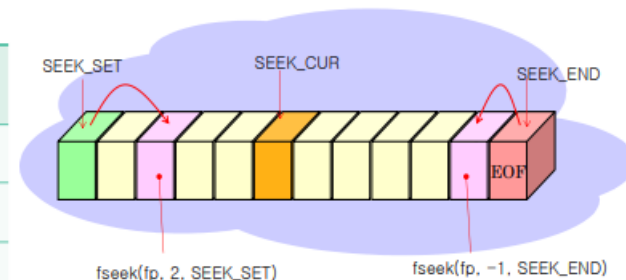
Syntax: fseek()

```
예 int fseek(FILE *fp, long offset, int origin);
```

FILE 포인터 거리 기준 위치

- fp는 파일 포인터
- offset은 기준 위치로부터 위치 표시자가 이동하는 거리를 나타냄
 - ✓ offset이 양수이면 앞으로 가고 음수이면 뒤로 감
- origin은 위치 표시자를 이동시키는 기준 위치를 나타냄

상수	값	설명
SEEK_SET	0	파일의 시작
SEEK_CUR	1	현재 위치
SEEK_END	2	파일의 끝



4. 파일 임의 접근

■ 파일 커서

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

int main(void)
{
    FILE* fp = fopen("test.txt", "r");

    int temp;
    fscanf(fp, "%d", &temp);
    fscanf(fp, "%d", &temp);

    int cur = ftell(fp);
    printf("%d\n", cur);

    rewind(fp);
    fscanf(fp, "%d", &temp);
    printf("%d\n", temp);

    fseek(fp, cur, SEEK_SET);
    fscanf(fp, "%d", &temp);
    printf("%d\n", temp);

    return 0;
}
```

test.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보...

15
235
12358
123
5
8
98
53
48

Microsoft Visu.

7
15
12358

5. 실습 종합 문제

■ 실습 3

- 구조체를 파일에 저장해보자. 구조체는 이진 파일의 형태로 저장되어야 한다.
- 사용자로부터 학생 이름, 학생 성적을 10개 입력 받아서 구조체 배열에 담고, 이를 파일에 저장하는 프로그램을 개발한다.
- 저장된 파일에서 학생 구조체 데이터를 읽고, 성적 순으로 정렬해서 출력하는 프로그램을 개발.
- 총 2개의 프로그램을 개발한다.

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

typedef struct _MyStruct {
    int id;
    char name[10];
} MyStruct;

int main(void) {
    FILE* fp = fopen("test.bin", "w");

    MyStruct test = { 4901, "Hello" };

    fwrite(&test, sizeof(MyStruct), 1, fp);

    return 0;
}
```

5. 실습 종합 문제

■ 실습 4

- 텍스트 파일에 세 와이파이의 신호 강도가 저장되어 있다.
- 와이파이는 각각 "AP_W24", "AP_K35", "AP_V72"라는 이름을 가지고 있다.
- 텍스트 파일에 아래처럼 와이파이 이름과 신호 강도가 무작위로 저장되어 있을 때, 와이파이 각각의 신호 강도 평균을 출력하는 프로그램을 개발해보자.

wifi_signal.csv - Windows 메!

파일(F) 편집(E) 서식(O) 보기

```
AP_K35,-44
AP_W24,-93
AP_K35,-51
AP_V72,-76
AP_W24,-69
AP_W24,-60
AP_W24,-28
AP_K35,-23
AP_V72,-74
AP_K35,-56
```