C 프로그래밍 및 실습

p.17 [11]

14. 파일 입출력

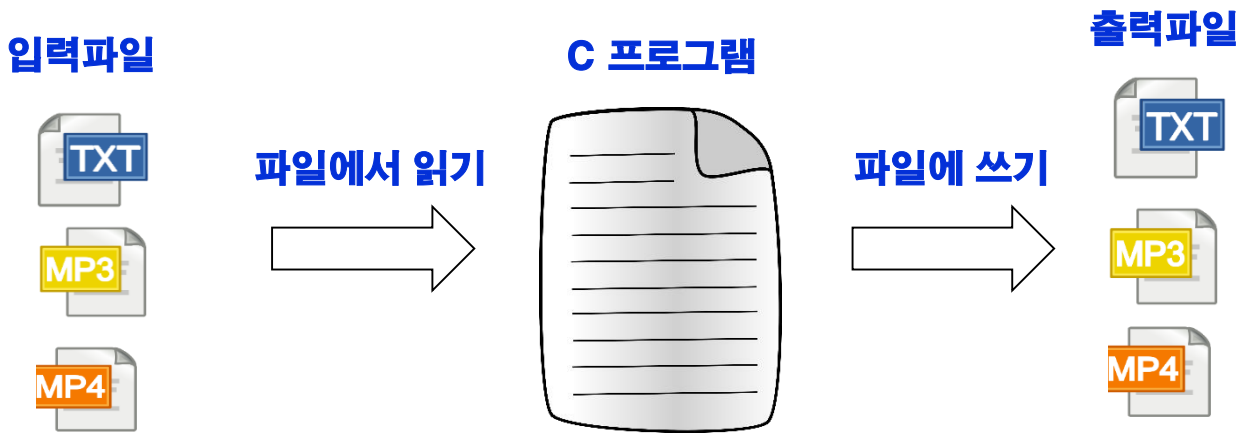
세종대학교

목차

1) 파일 입출력 개요

- **지금까지 사용한 입출력 방식**
  - 표준입력장치(키보드)를 통해 데이터를 입력하고,
    표준출력장치 (모니터)를 통해 결과를 보여줌
  - 프로그램 실행 때마다 입력, 출력은 프로그램 종료시 사라짐

- **프로그램의 실행/종료 여부와 무관하게 데이터를 보존하고 싶다면??**
  - 데이터를 **파일**로부터 입력받고, 결과를 **파일**로 저장(**파일 입출력**)



1) 파일 입출력 개요

- **파일 저장 방식에 따른 구분**

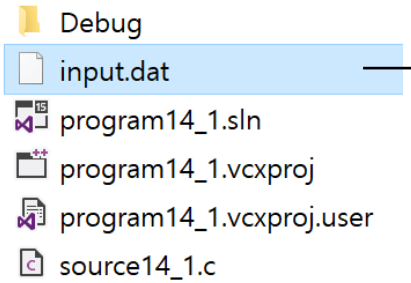
	텍스트(text) 파일	이진(binary) 파일
특성	✓사람이 인식할 수 있는 문자를 담고 있는 파일 ✓특별한 응용 프로그램 없이 내용 볼 수 있음 ✓메모장 프로그램을 통해 파일을 열었을 때, 읽을 수 있는 문자들로 표현됨 ✓모든 데이터가 문자열로 변환되어 기록됨	✓컴퓨터가 인식할 수 있는 데이터를 담고 있는 파일 ✓특정 응용 프로그램을 이용해야 액세스할 수 있는 파일(예, ppt 파일) ✓메모장 프로그램을 통해 파일을 열었을 때, 알아볼 수 없는 이상한 문자들로 표시 ✓수치 데이터가 문자로 변환되지 않고 곧바로 수치로 저장 ✓텍스트 파일보다 저장 공간을 적게 차지 ✓읽고 쓰기가 빠름

1) 파일 입출력 개요

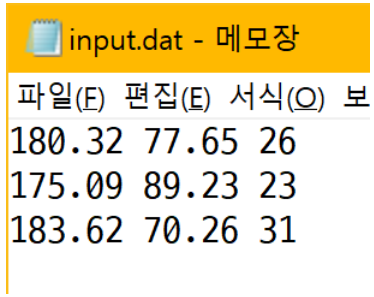
- **파일 입출력 따라 해보기**

  ✓ input.dat 파일로부터 키, 몸무게, 나이 정보를 입력받아

  ➔ 이 내용을 그대로 output.dat 파일에 출력하기



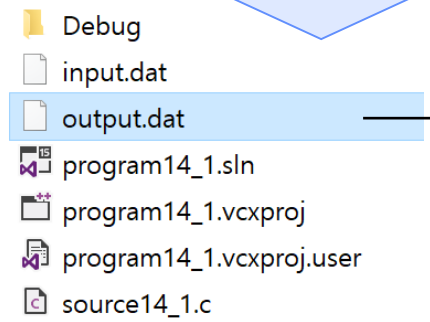
<현재 작업 폴더 내용 – 실행 전>



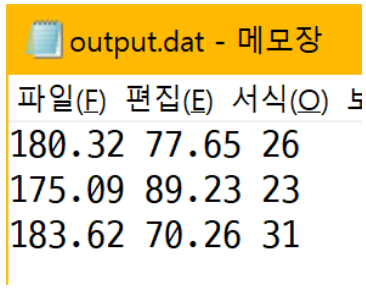
메모장으로 작성

<input.dat 파일의 내용>

(새로 생성)



<현재 작업 폴더 내용 – 실행 후>



<output.dat 파일의 내용>

1) 파일 입출력 개요

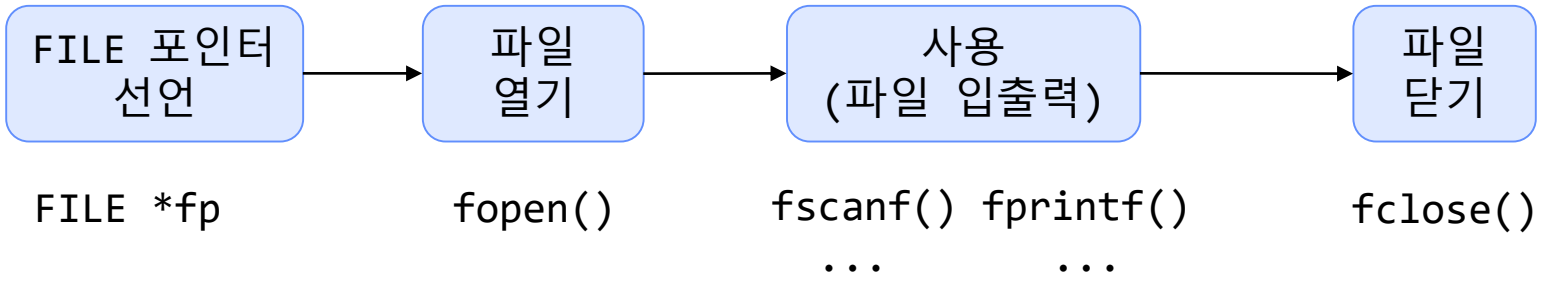
- **예제 프로그램 14.1**

```
#include <stdio.h>

int main() {
   double height, weight;
   int age, i;
   FILE *fp1, *fp2;                          // FILE 구조체 포인터 선언

   fp1 = fopen("input.dat", "r");            // input.dat 파일 열기
   fp2 = fopen("output.dat", "w");           // output.dat 파일 열기

   for (i = 0; i < 3; i++)   {
      fscanf(fp1, "%lf %lf %d", &height, &weight, &age); // 입력 받기
      fprintf(fp2, "%.2f %.2f %d\n", height, weight, age); // 출력하기
   }

   fclose(fp1);                                 // input.dat 파일 닫기
   fclose(fp2);                                 // output.dat 파일 닫기
   return 0;
}
```

목차

2) 파일 입출력 절차

- **파일 입출력 절차**
  - 어떤 파일에 입출력할 지에 대한 명시 필요
  - 필요한 자료형 및 함수는 <stdio.h>에 선언



2) 파일 입출력 절차

- **FILE 포인터 선언**
  - **FILE**은 파일 입출력 시 필요한 정보를 담은 구조체
  - 파일 입출력 시 각 파일마다 하나의 FILE 포인터를 연결하여 사용
  - 선언 형식

```
FILE *fp;   ⇨ 주의 - FILE은 반드시 대문자!
```

  - 표준 스트림
    - ✓ 표준 입출력 장치(키보드, 모니터)도 논리적으로 파일로 간주
    - ✓ 표준 스트림에 대한 FILE 포인터 명 (정해져 있음)

FILE 포인터 이름	스트림	의미
stdin	표준 입력 스트림	키보드로부터 입력 받음
stdout	표준 출력 스트림	모니터로 결과 출력
stderr	표준 오류 출력 스트림	모니터로 오류 메시지 출력

2) 파일 입출력 절차

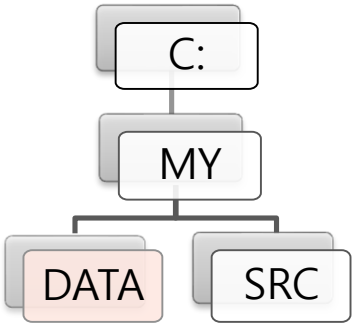
- **파일 열기: fopen() 함수**
  - 해당 파일에 대한 연결 요청을 의미
  - 해당 파일을 사용할 수 있도록 파일 포인터를 반환

함수원형	`FILE *fopen(char *filename, char *filemode);`	
함수인자	filename	연결 할 파일 이름
	filemode	파일 접근 방식에 따른 모드
반환값	✓ 파일열기에 성공 → FILE 포인터를 반환 ✓ 파일열기에 실패 → NULL을 반환	

```
1) FILE *fp = fopen("test.txt", "r");
2) FILE *fp2 = fopen("C:\\MY\\DATA\\test.txt", "a");
3) FILE *fp3 = fopen("..\\DATA\\test.txt", "w");
4) FILE *fp4 = fopen("DATA\\test.txt", "r");
```

2) 파일 입출력 절차

- **함수인자 filename**
  - 특정한 위치를 지정하지 않는 경우 → 현재 작업 폴더
    - ✓ 현재 작업 폴더 = (Visual studio에서는) 현재 소스 프로그램이 위치한 폴더
    - ✓ 예) `fopen("test.txt", "r");`



< 예시 폴더 구조 >

  - 절대경로
    - ✓ 드라이브 명부터 해당 파일이 있는 위치까지 전체 경로
    - ✓ 예) `fopen("C:\\MY\\DATA\\test.txt", "a");`

  - 상대경로
    - ✓ 현재 작업 폴더를 기준으로 해당 파일이 있는 위치까지의 경로
    - ✓ 예) `fopen("..\\DATA\\test.txt", "w");  // ..은 상위 폴더`
    - ✓ 예) `fopen("DATA\\test.txt", "r");`

2) 파일 입출력 절차

- **함수인자 filemode**
  - 개방할 파일의 용도에 따라 적합하게 지정해야 함
    - ✓ 적합한 모드 지정은 파일을 잘못 사용하는 것을 막을 수 있음

< 파일 접근 방식에 따른 모드 구분>

텍스트 모드	이진 모드	기능	설명
r	rb	읽기 전용 (read)	✓ 파일을 열 수 없는 경우 → NULL을 반환
w	wb	쓰기 전용 (write)	✓ 파일이 없는 경우 → 새로 빈 파일을 생성 ✓ 같은 이름의 파일이 존재하는 경우 → 해당 파일의 내용 삭제하고 새로 파일 생성
a	ab	추가 전용 (append)	✓ 파일이 없는 경우 → 새로 빈 파일을 생성 ✓ 같은 이름의 파일이 존재하는 경우 → 기존 파일의 마지막 부분에 내용을 추가

✓ 참고) '+' 기호를 붙이면 (예: r+, w+, ab+ 등) 수정 모드(읽기 쓰기 모두 가능)

2) 파일 입출력 절차

- **fopen() 함수 사용 시 주의사항**
  - fopen() 함수 호출 시, 이 함수의 반환 값을 반드시 검사하여 파일이 정상적으로 열렸는지 확인해야 함

```
FILE *fp = fopen("input.dat", "r");
if (fp == NULL)  {        // 파일 열기에 실패한 경우
     printf("Couldn't open file");   // 오류 처리 코드
     return -1;
}
```

2) 파일 입출력 절차

- **파일 입출력 함수**

처리 대상	처리 단위	파일 입력	파일 출력	비고
텍스트 파일	문자	fgetc()	fputc()	14.3절
	문자열	fgets()	fputs()	
	지정 형식	fscanf()	fprintf()	
이진 파일	블록	fread()	fwrite()	14.4절

2) 파일 입출력 절차

- **파일 닫기: fclose() 함수**
  - 현재 열린 파일과 FILE포인터와의 연결을 해제

함수 원형	**int fclose(FILE *fp);**	
함수 인자	fp	파일 포인터 변수명
반환 값	✓ 파일 닫기에 성공 → 0을 반환 ✓ 파일 닫기에 실패 → EOF를 반환	

```
FILE *fp = fopen("test.dat", "r");
if (fp == NULL) {
        printf("파일열기에 실패했습니다!\n");
        return -1;
}
fclose(fp);
```

2) 파일 입출력 절차

- **예제 프로그램 14-2**
  - 예제 프로그램 14-1 (슬라이드 p. 6)을 다음과 같이 바꿔보자.
    - ✓ 파일명을 사용자로부터 입력
      - ✓ 파일 이름을 저장할 문자 배열 필요
    - ✓ 정상적으로 열렸는지 확인

2) 파일 입출력 절차

- **예제 프로그램 14-2 (주요 변경 부분만)**

```
char fn1[10] = {'\0'}, fn2[11] = {'\0'}; // 파일 이름 저장 배열
...

printf("Input filename: ");
scanf("%s", fn1);                // 사용자로부터 입력 파일명을 입력받음
printf("Output filename: ");
scanf("%s", fn2);                // 사용자로부터 출력 파일명을 입력받음

fp1 = fopen(fn1, "r");           // 입력 파일 열기
if (fp1 == NULL)   {             // 입력 파일 열기의 성공 여부 판단
   printf("Couldn't open file");
   return -1;
}
fp2 = fopen(fn2, "w");          // 출력 파일 열기
if (fp2 == NULL)   {             // 출력 파일 열기의 성공 여부 판단
   printf("Couldn't open file");
   return -1;
}
...
```

목차

3) 텍스트 파일 입출력

- **지정 형식 파일 입력 함수: fscanf()**
  - 형식을 지정하여 파일의 데이터를 읽기 위한 함수
  - 여러 형태의 자료들(정수, 문자, 문자열 등)을 한번에 입력 가능
  - 함수의 첫 번째 인자로 파일 포인터가 사용된다는 것을 제외하고는 scanf() 함수와 사용법 동일

함수원형	`int fscanf(FILE *fp, char *format, …);`	
함수인자	`fp`	파일 포인터 변수명
	`format`	형식 제어 문자열
	…	입력하고자 하는 변수리스트
반환 값	✓ 성공 → 입력한 변수의 개수를 반환 ✓ 읽기 실패 → EOF를 반환	

3) 텍스트 파일 입출력

- **fscanf() 함수 사용 예**

```
char name[10] = {'\0'};
double weight;
int age;

FILE *fp = fopen("input.dat", "r");

fscanf(fp, "%s %d %lf", name, &age, &weight);
          // input.dat 파일로부터 데이터를 입력받음
fscanf(stdin, "%s %d %lf", name, &age, &weight);
          // 키보드로부터 데이터를 입력받음
...
```

✓ 첫 번째 인자를 stdin으로 지정하면, 표준입력으로부터 입력

✓ scanf() 함수를 사용한 것과 동일한 기능

✓ stdin, stdout, stderr는 따로 open, close 할 필요 없음 (자동 수행)

3) 텍스트 파일 입출력

- **지정 형식 파일 출력 함수: fprintf()**
  - 지정한 형식에 맞추어 파일로 출력
  - 함수의 첫 번째 인자로 파일 포인터가 사용된다는 것을 제외하고는 printf() 함수와 사용법 동일

함수원형	**int fprintf(FILE *fp, char *format, ...);**	
함수인자	fp	파일 포인터 변수명
	format	형식 제어 문자열
	...	출력하고자 하는 변수리스트
반환 값	✓ 성공 → 출력한 데이터의 바이트 수 ✓ 실패 → 음수를 반환	

3) 텍스트 파일 입출력

- **fprintf() 함수 사용 예**

```
char name[10] = "Tommy";
double weight = 78.45;
int age = 25;

FILE *fp = fopen("output.dat", "w");

fprintf(fp, "%s %d %.2lf", name, age, weight);
            // output.dat 파일로 데이터를 출력

fprintf(stdout, "%s %d %.2lf", name, age, weight);
            // 모니터로 데이터를 출력...
```

3) 텍스트 파일 입출력

- **문자 단위 파일 입출력 함수: fgetc() 와 fputc()**
  - 문자 한 개를 파일로부터 읽고 쓰기 위한 함수

함수 원형	`int fgetc(FILE *fp);`	
함수 인자	fp	파일 포인터 변수명
반환 값	✓ 성공 → 읽은 문자 반환 ✓ 실패(파일 끝 또는 오류 발생) → EOF 반환	

함수 원형	`int fputc(int char, FILE *fp);`	
함수 인자	char	출력하고자 하는 문자 상수 또는 변수
	fp	파일 포인터 변수명
반환 값	✓ 성공 → 출력하는 문자 char를 반환 ✓ 실패 → EOF 반환	

3) 텍스트 파일 입출력

- **프로그램 14-3: 문자 단위의 파일 입출력**
  1. 'x'가 입력될때까지 문자를 반복적으로 입력 받아 example.txt 파일에 문자 단위로 출력

  2. example.txt 파일의 내용을 화면에 문자 단위로 출력

3) 텍스트 파일 입출력

- **프로그램 14-3: 문자 단위의 파일 입출력 (주요 코드)**

```
char fn[] = "example.txt";
FILE *fp = fopen(fn, "w");          // 출력 파일 열기
...
ch = getchar();                     // 문자 하나를 표준입력으로 입력
while (ch != 'x') {                 // 'x'가 입력될 때까지 반복
   fputc(ch, fp);                   // 출력 파일로 문자 한 개 출력
   ch = getchar();                  // 키보드로부터 문자 한 개를 입력
}
fclose(fp);                         // 출력 파일 닫기
...
fp = fopen(fn, "r");                // 입력 파일 열기
...
ch = fgetc(fp);                     // 입력파일에서 문자 한 개 읽음
while (!feof(fp)) {                 // 파일의 끝까지 반복 (추후 학습)
   putchar(ch);                     // 읽은 문자를 모니터로 출력
   ch = fgetc(fp);                  // 입력파일에서 문자 한 개 읽음
}
fclose(fp);                         // 입력 파일 닫기
```

3) 텍스트 파일 입출력

- **문자열 단위의 파일 입력 함수: fgets( )**
  - 파일에 쓰여진 문자열을 읽는데 사용하는 함수

함수 원형	**char *fgets(char *s, int n, FILE *fp);**	
함수 인자	s	파일로부터 읽은 문자열을 저장할 포인터
	n	저장할 문자의 최대 개수 (널 문자 포함)
	fp	파일 포인터 변수명
반환 값	✓ 성공 → 문자열 s를 반환 ✓ 파일의 끝에 도달하거나 실패/오류 발생 → NULL을 반환	

3) 텍스트 파일 입출력

- **fgets()의 기능**
  - 다음 조건이 만족될 때까지 파일 fp로 부터 문자를 읽어서 s가 가리키는 곳에 저장
    - ✓ n-1 개의 문자(공백, 탭 문자 포함)를 읽거나,
    - ✓ 개행 문자를 만나거나,
    - ✓ 파일의 끝에 도달
    - ✓ * 문자열 마지막에 널 문자 추가
  - 다른 입력 함수와 다른 점
    - ✓ fscanf와 달리, 공백, 탭 문자를 만나도 계속 읽음
    - ✓ 개행 문자도 문자열에 저장

3) 텍스트 파일 입출력

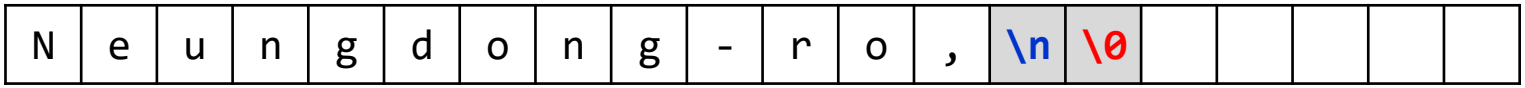
info.txt

```
Neungdong-ro,↵
Gwangjin-gu, Seoul, Korea.↵
```

- **fgets() 함수 사용 예**

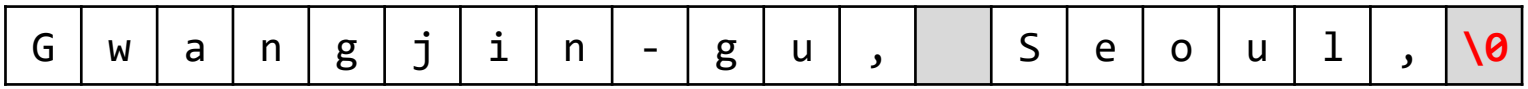
1) fgets(str1, **20**, fp);

✓ 중간에 개행 문자를 만나 읽기 종료. 마지막에 널 문자 추가



2) fgets(str2, **20**, fp);

✓ **19**개의 문자를 읽은 후 뒤에 널 문자를 합쳐 str2 배열에 저장



3) fgets(str3, **20**, fp);

✓ 이전에 읽은 곳 다음부터 읽다가 19개의 문자를 읽기 전에 개행 문자를 읽게 되어, 그 뒤에 널 문자를 합쳐 str3 배열에 저장

3) 텍스트 파일 입출력

- **문자열 단위의 파일 출력 함수: fputs()**
  - 파일에 문자열을 출력하는 함수
  - 문자열의 끝을 나타내는 널 문자는 파일에 쓰지 않으며, 그 뒤에 개행 문자도 자동으로 들어가지 않음
    - ✓ (비교) puts( ) 함수는 문자열 출력 후 자동으로 개행

함수 원형	**int fputs(char *str, FILE *fp);**	
함수 인자	str	출력하고자 하는 문자열
	fp	파일 포인터 변수명
반환 값	✓ 성공 → 출력한 바이트 수를 반환 ✓ 실패/오류발생 → EOF를 반환	

3) 텍스트 파일 입출력

- **프로그램 14-4: 문자열 단위 파일 입출력**
  1. 표준입력으로부터 국가명을 입력 받고,
     이를 Cnty_list.txt 파일에 저장

  2. Cnty_list.txt 파일의 내용을 화면에 문자열 단위로 출력

3) 텍스트 파일 입출력

- **프로그램 14-4: 문자열 단위 파일 입출력 (주요 코드)**

```
char fname[] = "Cnty_list.txt", temp[50] = { '\0' };
FILE *fp = fopen(fname, "w");     // 출력 파일 열기
...
for (i = 0; i < 4; i++) {
   fgets(temp, 50, stdin);  // 문자열을 입력받아 temp에 저장
   fputs(temp, fp);         // temp에 저장된 문자열을 파일에 출력
}
fclose(fp);                   // 출력 파일 닫기

fp = fopen(fname, "r");       // 입력 파일 열기
...
for (i = 0; i < 4; i++) {
   fgets(temp, 50, fp);  // 입력파일로부터 문자를 읽어 temp에 저장
   printf("%s", temp);   // temp에 저장된 문자열을 모니터로 출력
}
fclose(fp);                   // 입력 파일 닫기
```

3) 텍스트 파일 입출력

- **파일의 끝 확인하기: feof( )**
  - 파일의 끝까지 데이터를 모두 읽어 들인 상태인지를 확인

함수 원형	**int feof(FILE *fp);**	
함수 인자	fp	파일 포인터 변수명
반환 값	✓ 파일의 끝이면 → 0이 아닌 값을 반환 ✓ 파일의 끝이 아니면 → 0를 반환	

```
FILE *fp = fopen("test.txt", "r");
char str[50]= {'\0’};

fgets(str, sizeof(str), fp); // 데이터 읽은 후, feof 호출
while( !feof(fp) ) {          // 파일 끝이 아닌 동안
   fgets(str, sizeof(str), fp);
   printf("%s", str);
}
```

3) 텍스트 파일 입출력

- **EOF (End Of File)란?**
  - 파일의 끝을 표현하기 위해 정의해 놓은 상수 (즉, -1)
  - EOF도 파일의 내용으로 간주되므로 주의

- **EOF vs. feof( ) 함수**
  - 모든 파일의 끝에는 EOF가 존재 → EOF는 파일의 한 부분임
  - feof( ) 함수는 EOF를 읽으면, **0을 반환!**
  - **그 이후에 0이 아닌 값을 반환!**

➔ 파일로부터 **데이터를 먼저 읽은 후에** feof() 함수 사용
앞 슬라이드의 예제 및 프로그램 14.3 참조

3) 텍스트 파일 입출력

- **feof() 함수 호출 먼저 하면?**
  - "test.txt" 파일을 빈 파일로 하여, 실행시켜 보자. 결과는?
  - "test.txt" 파일에 데이터를 저장한 후, 실행시켜 보자. 결과는?

```
FILE *fp = fopen("test.txt", "r");
char str[50]= {'\0'};

~~fgets(str, sizeof(str), fp);~~ // 지우고 실행시켜보자.

while( !feof(fp) ) {          // 파일 끝이 아닌 동안
   fgets(str, sizeof(str), fp);
   printf("%s", str);
}
```

목차

4) 이진 파일 입출력 (심화 내용)

- **블록 단위의 파일 입출력 함수**
  - 이진 파일은 바이트 단위의 연속된 데이터 집합인 **블록 단위로** 데이터를 저장하고 읽음
  - 일정한 크기의 데이터를 한 번에 읽거나 쓸 수 있음
  - 이진 파일의 데이터를 읽을 때에는 **fread()** 함수를, 데이터를 저장할 때에는 **fwrite()** 함수를 사용

  - 파일에 순차적으로 접근할 수 있을 뿐만 아니라(순차 접근) 임의의 위치로 곧바로 이동하여 읽고 쓸 수도 있음(임의 접근)

4) 이진 파일 입출력 (심화 내용)

- **블록 단위 파일 입력: fread()**
  - 이진파일에서 (블록 크기 x 블록 개수) 만큼의 데이터를 읽음

함수 원형	`unsigned int fread(void *buf, unsigned int size, unsigned int count, FILE *fp);`	
함수 인자	buf	읽은 데이터를 저장할 버퍼의 시작 주소
	size	읽어올 데이터 블록 하나의 크기 (바이트)
	count	읽어올 데이터 블록의 개수
	fp	파일 포인터 변수명
반환 값	✓ 성공 → 파일로부터 읽은 블록의 개수를 반환 ✓ 파일의 끝 혹은 실패 → count 보다 작은 값을 반환	

4) 이진 파일 입출력 (심화 내용)

- **fread() 함수 사용 예**

```
int age, ID[10];
FILE *fp = fopen("input.dat", "rb");

fread(&age, sizeof(int), 1, fp);
                // fp에 연결된 이진 파일로부터
                // 정수(int) 1개를 읽어 age 변수에 저장

fread(ID, sizeof(int), 10, fp);
                // fp에 연결된 이진 파일로부터
                // 정수(int) 10개를 읽어 ID 배열에 저장
```

4) 이진 파일 입출력 (심화 내용)

- **블록 단위 파일 출력: fwrite()**
  - 이진파일에 (블록 크기 x 블록 개수) 만큼의 데이터를 출력

함수 원형	**unsigned int fwrite(const void *buf, unsigned int size, unsigned int count, FILE *fp);**	
함수 인자	buf	출력할 데이터를 저장하고 있는 버퍼의 시작 주소
	size	출력할 데이터 블록 하나의 크기 (바이트)
	count	출력할 데이터 블록의 개수
	fp	파일 포인터 변수명
반환 값	✓ 성공 → 파일에 출력한 블록의 개수를 반환 ✓ 실패 → count 보다 작은 값을 반환	

4) 이진 파일 입출력 (심화 내용)

- **fwrite() 함수 사용 예**

```
int age, ID[10];
FILE *fp = fopen("output.dat", "wb");

fwrite(&age, sizeof(int), 1, fp);
                // age 변수에 저장된 정수 1개를
                // fp에 연결된 이진 파일에 저장

fwrite(ID, sizeof(int), 10, stdout);
                // ID 배열에 저장된 정수 10개를
                // 표준출력장치에 출력
```

4) 이진 파일 입출력 (심화 내용)

- **프로그램 14-5: 블록 단위 파일 입출력**

1. 표준입력으로부터 학생이름, 중간성적, 기말성적을 입력받아 이진파일에 저장
   - ✓ 학생 정보 구조체를 정의하여 사용

```
typedef struct person_score {
        char name[10];    double mid;        double final;
}person;                                  // 학생 정보
```

   - ✓ EOF 가 입력될 때까지 반복
     - ✓ 윈도우 운영체제: ^Z(ctrl+Z)
     - ✓ 유닉스/리눅수 운영체제: ^D(ctrl+D)

2. 생성된 파일로부터 저장된 내용을 읽어 표준출력으로 출력

4) 이진 파일 입출력 (심화 내용)

- **프로그램 14-5: 블록 단위 파일 입출력 (주요코드 1/2)**

```
person ps;
char fname[] = "score.dat", tmp[80] = { '\0' };

FILE *fp = fopen(fname, "wb");   // 출력 파일 열기
...

/* 사용자로부터 구조체 멤버 값을 입력받아, 파일에 저장하기 */
fscanf(stdin, "%s %lf %lf", ps.name, &ps.mid, &ps.final);    // 입력
while (!feof(stdin)) {            // EOF가 입력될 때까지 반복
   fwrite(&ps, sizeof(person), 1, fp);   // 파일에 저장
   fscanf(stdin, "%s %lf %lf", ps.name, &ps.mid, &ps.final); // 입력
}

fclose(fp);                 // 출력 파일 닫기
(다음 슬라이드에 계속)
```

4) 이진 파일 입출력 (심화 내용)

- **프로그램 14-5: 블록 단위 파일 입출력 (주요코드 2/2)**

```
fp = fopen(fname, "rb");                    // 입력 파일 열기
...

/* 파일로부터 구조체 각 멤버 값을 읽어, 화면에 출력하기 */
fread(&ps, sizeof(person), 1, fp);  // 파일로부터 구조체 읽기
While (!feof(fp)) {                  // EOF가 입력될 때까지 반복
   fprintf(stdout, "%s %.2lf %.2lf\n",
                  ps.name, ps.mid, ps.final); //  화면에 출력
   fread(&ps, sizeof(person), 1, fp);    // 파일로부터 구조체 읽기
}

fclose(fp);      // 입력 파일 닫기
```

4) 이진 파일 입출력 (심화 내용)

- **파일의 임의 접근 처리 방식**
  - 이진파일은 **임의의 위치에서 바로 읽기/쓰기**를 할 수 있는 임의 접근 방식으로 파일 입출력 가능
  - 파일 위치 지시자를 조작하는 함수를 사용
    - ✓ 파일 위치 지시자: 파일에서 읽거나 쓸 데이터의 위치를 나타냄
  - fseek(), ftell(), rewind() 함수 사용

4) 이진 파일 입출력 (심화 내용)

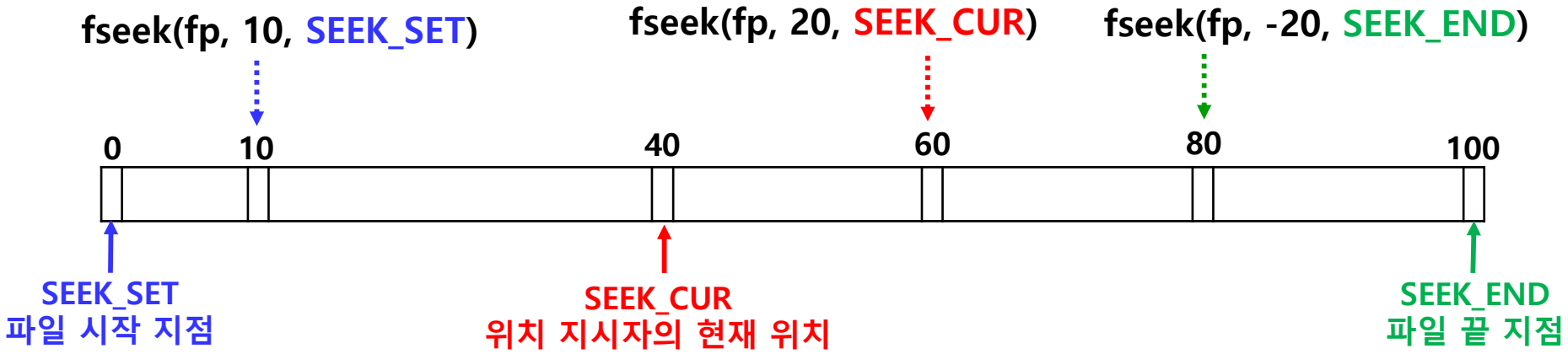
- **fseek() 함수**
  - 파일 위치 지시자를 지정한 위치로 이동시킬 수 있는 함수
  - origin으로부터 offset만큼 떨어진 곳으로 이동

함수 원형	**int fseek(FILE *fp, long int offset, int origin);**	
함수 인자	fp	파일 포인터 변수명
	offset	✓ origin으로부터 이동할 바이트 수 - 양수(+): 순방향(기준점 이후) - 음수(-): 역방향(기준점 이전)
	origin	✓ offset을 적용할 기준점 - SEEK_SET(0): 파일의 맨 처음 위치 - SEEK_CUR(1): 파일에서의 현재 위치 - SEEK_END(2): 파일의 맨 끝 위치
반환 값	✓ 성공 → 0을 반환 ✓ 실패 → 0이 아닌 값을 반환	

4) 이진 파일 입출력 (심화 내용)

- **fseek() 함수 사용 예**

```
fseek(fp, 10, SEEK_SET);
   → 위치 지시자를 파일의 시작 지점에서 10바이트 뒤로 이동

fseek(fp, 20, SEEK_CUR);
   → 위치 지시자를 현재 위치에서 20바이트 뒤로 이동

fseek(fp, -20, SEEK_END); // 2
   → 위치 지사자를 파일의 끝 지점에서 20바이트 앞으로 이동
```



4) 이진 파일 입출력 (심화 내용)

- **ftell() 함수**
  - 현재 파일 위치 지시자가 가리키는 곳의 위치를 반환하는 함수
  - 현재 위치가 파일의 시작 위치로부터 몇 바이트 떨어져 있는지를 알려줌

함수 원형	**long ftell (FILE *fp);**	
함수 인자	fp	파일 포인터 변수명
반환 값	✓ 성공 → 읽기/쓰기 위치를 반환 ✓ 실패/오류 → -1을 반환	

```
FILE *fp = fopen("input.dat", "rb");
long pos = ftell(fp);
```

4) 이진 파일 입출력 (심화 내용)

- **rewind() 함수**
  - 파일 위치 지시자를 파일의 시작 지점으로 이동시키는 함수
  - fseek(fp, 0, SEEK_SET)와 동일한 기능

함수 원형	**void rewind(FILE *fp)**	
함수 인자	fp	파일 포인터 변수명

```
char str[50] = {'\0'};
FILE *fp = fopen("input.dat", "rb");
fread(str, sizeof(char), 10, fp);
puts(str);
rewind(fp);   // 처음 위치로 되돌림
```