[13주차 2강]
파일 입출력(2)

학습 내용

- 14.2 파일 입출력 절차

학습 목표

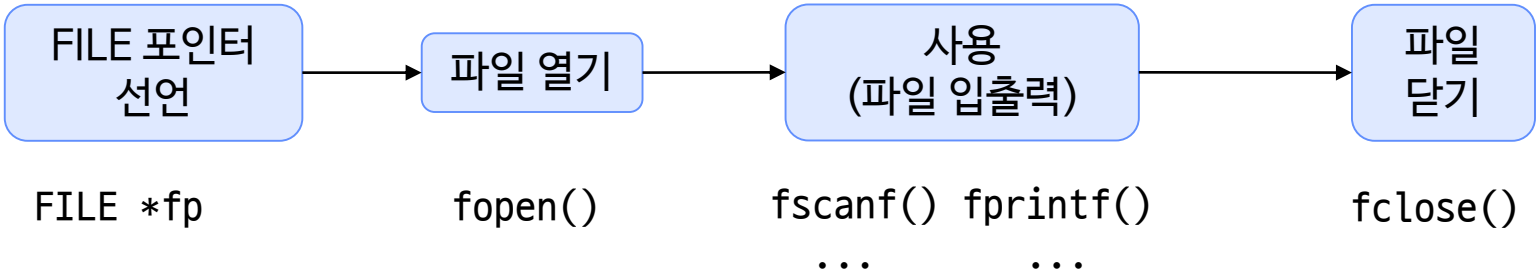
- 14.2 파일 입출력 절차를 이해한다.

C프로그래밍 및 실습

1. 파일 입출력 개요
2. **파일 입출력 절차**
3. 텍스트 파일 입출력

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

- FILE은 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!
```

2. 파일 입출력 절차

표준 스트림

- 표준 입출력 장치(키보드, 모니터)도 논리적으로 파일로 간주하여 처리
- 표준 스트림에 대한 FILE 포인터 명 (정해져 있음)
- 별도로 열거나 닫을 필요 없음 (자동 수행)

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

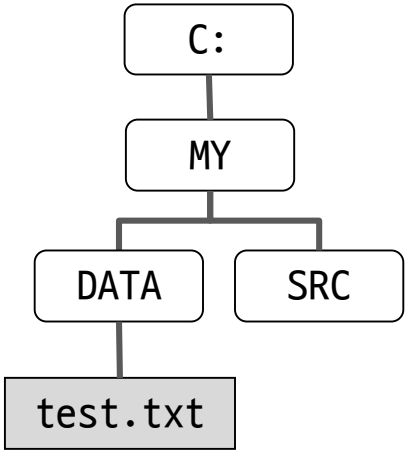
함수원형	`FILE *fopen(char *filename, char *filemode);`	
함수인자	filename	연결 할 파일 이름
	filemode	파일 접근 방식에 따른 모드
반환값	✓ 파일열기에 성공 → FILE 포인터를 반환 ✓ 파일열기에 실패 → NULL을 반환	

```
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

- 해당 파일의 위치와 이름 명시
- 위치는 상대경로 또는 절대경로 표현 (다음 슬라이드)
  - ✓ 예) `fopen("test.txt", "r");`
  - ✓ 예) `fopen("C:\\MY\\DATA\\test.txt", "a");`
  - ✓ 예) `fopen("..\\DATA\\test.txt", "w");`
- (주의) 경로에서 역슬래쉬 \ 를 표현하기 위해 \\ 을 사용 (2.5절 특수 문자 참고)



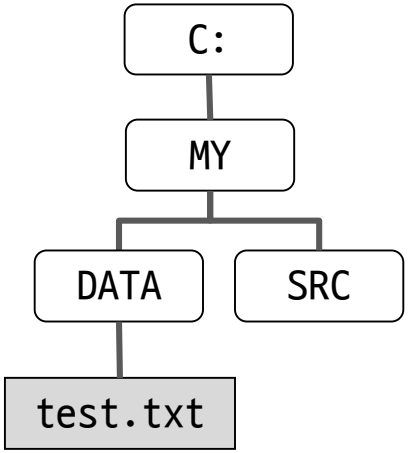
〈 예시 폴더 구조 〉

2. 파일 입출력 절차

폴더의 경로 표현 방법

- 절대경로
  - ✓ 드라이브 명부터 해당 파일이 있는 위치까지 전체 경로
  - ✓ 예) C:\MY\DATA\test.txt
  - ✓ Unix 또는 Linux에서는 \ 대신 / 사용

- 상대경로
  - ✓ 현재 작업 폴더를 기준으로 해당 파일이 있는 위치까지의 경로
  - ✓ 예) DATA\test.txt (현재 작업 폴더가 MY 인 경우)
  - ✓ 예) ..\DATA\test.txt (현재 작업 폴더가 SRC 인 경우)
  - ✓ ..은 상위 폴더를 의미



〈 예시 폴더 구조 〉

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〉

텍스트모드	이진모드	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);
```

※ 실습하기

[예제 프로그램 14-2] 예제 프로그램 14-1 을 다음과 같이 바꿔보자.

- ✓ 파일명을 사용자로부터 입력
  - ✓ 파일 이름을 저장할 문자 배열 필요
- ✓ 정상적으로 열렸는지 확인

※ 실습하기

[예제 프로그램 14-2] (주요 변경 부분만)

```
char fn1[10] = {'\0'}, fn2[10] = {'\0'}; // 파일 이름 저장 배열
...
printf("Input filename: ");
scanf("%s", fn1);                // 사용자로부터 입력 파일명을 입력 받음
printf("Output filename: ");
scanf("%s", fn2);                // 사용자로부터 출력 파일명을 입력 받음
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;
}
...
```

학습 정리

- 파일 입출력을 위해서는 어떤 파일에 입출력할 지에 대한 명시가 필요하고, 파일을 열고 닫는 절차가 필요

- 파일 입출력에 필요한 자료형 및 함수는 **〈stdio.h〉**에 선언되어 있음

- 파일을 사용하기 위해서는 우선 **fopen()** 함수를 이용하여 파일을 열고, 사용 후에는 **fclose()** 함수를 이용하여 파일을 닫아야 함

- **FILE 포인터**는 파일에 접근하기 위한 자료형