

9장 파일입출력

한림대학교 소프트웨어융합대학 양은샘.



9장 파일 입출력

- 안녕하세요? 여러분!
- 이번 장에서는 파일 입출력에 대해 학습합니다.
- C 언어에서는 키보드 입력과 화면 출력 뿐만 아니라 파일에서 입력과 파일로 출력도 할 수 있습니다.
- 이번 장에서는 파일 입출력을 이용해서 다양한 입력과 출력을 구현 방법을 학습 합니다.
- 지난 시간에 학습한 내용을 리뷰한 후 학습을 시작하도록 하겠습니다.

지난 시간 Review

8.1 구조체 정의와 구조체 변수의 선언

8.2 구조체 변수의 배열과 포인터

8.3 중첩된 구조체

8.4 자기 참조 구조체

8.5 구조체와 함수

8.6 비트 필드 구조체

8.7 공용체

8.8 구조체와 공용체의 응용

☐ 개념 확인 학습

☐ 적용 확인 학습

☐ 응용 프로그래밍

9장 파일 입출력

9.1 파일 포인터

가. 파일 열기 : `fopen()`

나. 파일 닫기 : `fclose()`

9.2 형식 지정 텍스트 파일 입출력 : `fprintf()`, `fscanf()`

9.3 블록 단위 바이너리 파일 입출력 : `fwrite()`, `fread()`

9.4 파일의 끝과 오류 처리

9.5 랜덤 파일 처리

9.6 그 밖의 파일 입출력 함수들

9.7 파일 입출력 응용

☐ 개념 확인 학습

☐ 적용 확인 학습

☐ 응용 프로그래밍

학습 목표

- 파일 포인터의 의미를 이해한다.
 - 파일 입출력을 위해 파일을 여는 방법과 닫는 방법을 안다.
 - 텍스트 파일의 입출력을 할 수 있다.
 - 바이너리 파일의 블록 단위 입출력을 할 수 있다.
 - 파일과 관련된 함수를 사용할 수 있다.
 - 파일에서 랜덤한 영역을 접근 할 수 있다.
 - 구조체와 파일 입출력을 응용하여 다양한 조건의 문제들을 해결 할 수 있다.
-
- 개념 확인 학습으로 배운 내용을 정리한다.
 - 적용 확인 학습으로 개념 습득 여부를 확인한다.
 - 응용 프로그래밍으로 문제해결력을 키운다.

입력과 출력 형태

- 표준 입출력 도우미
 - 표준 입력(stdin) : 키보드
 - 표준 출력(stdout) : 화면
- 파일 입출력 도우미
 - 파일 포인터(File Pointer)
- 파일 입출력을 수행하려면
 - 가장 먼저 파일 포인터부터 만들어야 함.



[그림 9.1] 프로그램의 입력과 출력 형태

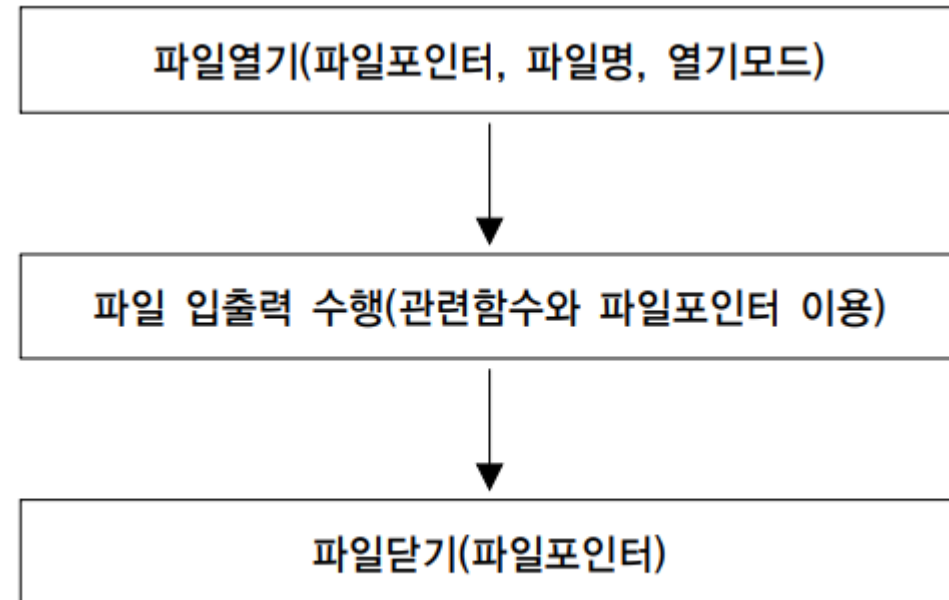
파일 포인터

➡ 파일 포인터 형식

FILE *파일포인터명;

➡ 파일 포인터 선언의 예

FILE *fp, *in, *out;



[그림 9.2] 파일 처리 과정

파일 열기 : fopen()

➡ fopen() 형식

```
filepointer = fopen(char *filename, char *openmode);
```

- filepointer : 파일 포인터 이름
- filename : 사용하고자 하는 파일의 이름
- openmode: 사용하고자 하는 파일을 여는 방법([표 9.1] 참조)
- 성공 : 파일 포인터를 반환, 실패 : NULL 값을 반환.

➡ fopen() 사용 예

```
FILE *fp, *in, *out;
```

```
char *filename = "ans.txt", *openmode = "w+b";
```

```
fp = fopen("test.txt", "r"); //파일이름으로 직접 열기
```

```
in = fopen("a:WworkWtest1.txt", "w+"); //파일이름과 경로 지정
```

```
out = fopen(filename, "a+"); //문자열 변수를 이용한 파일 열기
```

```
out = fopen(filename, openmode);
```

[표 9.1] fopen()에서 파일을 여는 방법

여는 방법	사용 예	기능
r	r, rt, rb, r+b	파일을 읽기 전용으로 연다. 해당 파일이 없으면 오류가 발생한다.
r+	r+t	존재하는 파일을 읽기와 쓰기가 가능한 상태로 연다. 해당 파일이 없으면 오류가 발생한다. 파일 포인터는 파일의 시작 부분에 위치한다.
w	wt, wb	파일을 쓰기 전용으로 연다. 파일은 자동으로 만들어 진다. 같은 이름의 파일이 이미 존재하면 내용을 모두 지우고 파일 포인터는 파일의 시작 부분에 위치한다.
w+	w+t, w+b	파일을 읽기와 쓰기가 가능한 상태로 연다. 파일은 자동으로 만들어 진다. 같은 이름의 파일이 이미 존재하면 내용을 모두 지우고 파일 포인터는 파일의 시작 부분에 위치한다.
a	at, ab	파일을 추가(쓰기) 전용으로 연다. 파일의 내용이 있는 경우 파일 포인터가 파일의 맨 끝으로 자동으로 이동해 새로운 내용을 파일의 마지막 부분부터 덧붙여 쓸 수 있도록 한다. 만약 해당 파일이 없으면 새로운 파일이 자동으로 생성된다.
a+	a+t, a+b	파일을 읽기와 추가(쓰기)가 가능한 상태로 연다. 지정된 파일이 없으면 생성되고, 있으면 그 파일의 맨 끝으로 파일포인터가 이동하여 읽기 또는 쓰기가 이루어진다.

파일 닫기 : fclose()

➡ fclose() 형식

```
int fclose(FILE *fp)
```

➡ fclose() 사용 예

```
FILE *fp;  
fclose(fp);
```

형식 지정 텍스트 파일 입출력 : fprintf(), fscanf()

⇒ fprintf() / fscanf() 형식

```
int fprintf(FILE *fp, “출력형식지정문자열”, 변수1, 변수2, ....);  
int fscanf(FILE *fp, “입력형식지정문자열”, 변수1, 변수2, ....);
```

⇒ fprintf() / fscanf() 사용 예

```
FILE *fp;  
int num;  
char name[10];
```

```
fprintf(fp, “%s %d”, name, num); //fp가 지시하는 파일로 출력  
fscanf(fp, “%s %d”, name, &num); //fp가 지시하는 파일에서 읽어 해당 변수  
에 저장.
```

```
fprintf(stdout, “%s %d”, name, num); //stdout 즉, 모니터로 출력  
fscanf(stdin, “%s %d”, name, &num); //stdin 즉, 사용자가 키보드로 입력하  
는 내용을 해당 변수에 저장.
```

fprintf() 함수의 활용

[예제 9.1] fprintf() 함수의 활용

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

```
#include <stdlib.h>
```

```
#define SIZE 4
```

```
struct unit {
```

```
    char fruit[10];
```

```
    int price;
```

```
};
```

```
typedef struct unit unit;
```

선택 명령 프롬프트

C:\lecture_src\ctest>ctest fruit.txt

apple 800

banana 500

orange 600

peach 1000

모니터에 출력된 내용이 파일에 저장되었습니다.
fruit.txt 파일을 열어보세요.

fruit - Windows 메모장

파일(F) 편집(E) 서식(O)

apple 800

banana 500

orange 600

peach 1000

```
int main(int argc, char *argv[])
```

```
{
```

```
    FILE *fp;
```

```
    unit list[SIZE] = { "apple", 800, "banana", 500, "orange", 600, "peach", 1000
```

```
};
```

```
    fp = fopen(argv[1], "w");
```

```
    //fp = fopen("fruit.txt", "w");
```

```
    if (fp == NULL) {
```

```
        //if (fopen_s(&fp, argv[1], "w")) {
```

```
        //if (fopen_s(&fp, "fruit.txt", "w")) {
```

```
            printf("%s Can't open file", argv[1]);
```

```
            exit(1);
```

```
    }
```

```
    for (int i = 0; i < sizeof(list) / sizeof(unit); i++) {
```

```
        fprintf(stdout, "%s %dWn", (list + i)->fruit, (list + i)->price);
```

```
        fprintf(fp, "%s %dWn", (list + i)->fruit, (list + i)->price);
```

```
    }
```

```
    fprintf(stdout, "Wn모니터에 출력된 내용이 파일에 저장되었습니다.Wn");
```

```
    fprintf(stdout, "%s 파일을 열어보세요.WnWn", argv[1]);
```

```
    fclose(fp);
```

```
    return 0;
```

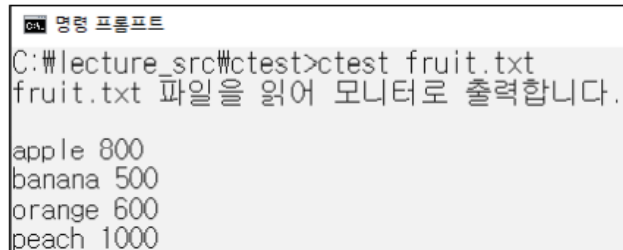
```
}
```

fscanf() 함수의 활용

[예제 9.2] fscanf() 함수의 활용

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4

struct unit {
    char fruit[10];
    int price;
};
typedef struct unit unit;
```



```
명령 프롬프트
C:\lecture_src\ctest>ctest fruit.txt
fruit.txt 파일을 읽어 모니터로 출력합니다.
apple 800
banana 500
orange 600
peach 1000
```

```
int main(int argc, char *argv[])
{
    FILE *fp;
    unit list[SIZE];

    fp = fopen(argv[1], "r");
    if (fp == NULL) {
        //if (fopen_s(&fp, argv[1], "r")) {
        printf("%s Can't open file", argv[1]);
        exit(1);
        }

    fprintf(stdout, "%s 파일을 읽어 모니터로 출력합니다.\n\n", argv[1]);

    for (int i = 0; i < sizeof(list) / sizeof(unit); i++) {
        fscanf(fp, "%10s %d", (list + i)->fruit, &(list + i)->price);
        fprintf(stdout, "%s %d\n", (list + i)->fruit, (list + i)->price);
    }
    fclose(fp);
    return 0;
}
```

fprintf()와 fscanf() 함수의 활용

[예제 9.3] fprintf()와 fscanf() 함수의 활용

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4
```

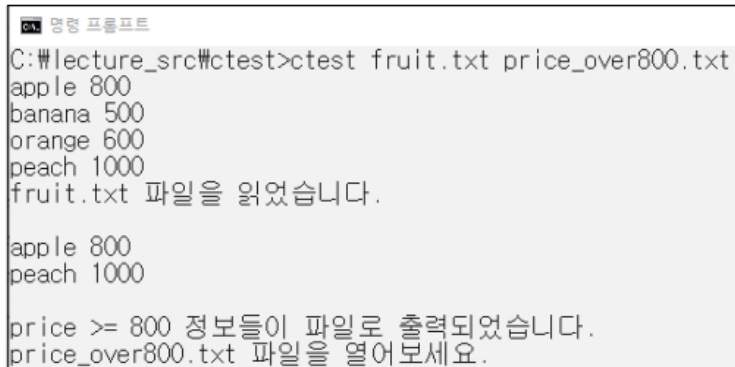
```
struct unit {
    char fruit[10];
    int price;
};
```

```
typedef struct unit unit;
```

```
int main(int argc, char *argv[])
{
```

```
    FILE *fp;
    unit list[SIZE];
```

```
    fp = fopen(argv[1], "r"); //argv[1] = "fruit.txt"
    if (fp == NULL) {
        printf("%s Can't open file", argv[1]);
        exit(1);
    }
```



명령 프롬프트

```
C:\lecture_src\ctest>ctest fruit.txt price_over800.txt
apple 800
banana 500
orange 600
peach 1000
fruit.txt 파일을 읽었습니다.

apple 800
peach 1000

price >= 800 정보들이 파일로 출력되었습니다.
price_over800.txt 파일을 열어보세요.
```



price_over800

```
apple 800
peach 1000
```

```
for (int i = 0; i < sizeof(list) / sizeof(unit); i++) {
    fscanf(fp, "%s %d", (list + i)->fruit, &(list + i)->price);
    fprintf(stdout, "%s %d\n", (list + i)->fruit, (list + i)->price);
}
```

```
fprintf(stdout, "%s 파일을 읽었습니다.\n\n", argv[1]);
fclose(fp);
```

```
fp = fopen(argv[2], "w"); //price_over800.txt
if (fp == NULL) {
    printf("%s Can't open file", argv[2]);
    exit(1);
}
```

```
for (int i = 0; i < sizeof(list) / sizeof(unit); i++) {
    if ((list + i)->price >= 800) {
        fprintf(stdout, "%s %d\n", (list+i)->fruit, (list+i)->price);
        fprintf(fp, "%s %d\n", (list+i)->fruit, (list+i)->price);
    }
}
```

```
fprintf(stdout, "\nprice >= 800 정보들이 파일로 출력되었습니다.\n");
fprintf(stdout, "%s 파일을 열어보세요.\n\n", argv[2]);
fclose(fp);
return 0;
```

블록 단위 바이너리 파일 입출력 : fwrite(), fread()

- 정수 한 개, 정수 여래 개 등 특정 크기의 데이터를 입출력 해야 할 경우
 - .data .bin 등의 이진 모드 파일 확장자 사용

➡ fwrite() / fread() 형식

```
size_t fwrite(const void *buffer, size_t size, size_t count, FILE *fp);
```

```
size_t fread(void *buffer, size_t size, size_t count, FILE *fp);
```

- `size_t fwrite(버퍼, 블록 크기, 블록 수, 파일 포인터);`
- `size_t fread(버퍼, 블록 크기, 블록 수, 파일 포인터);`
- `buffer`가 가리키는 곳에서 `size` 바이트 크기를 갖는 `count`개의 객체를 파일 포인터 `fp`와 관련된 파일에 쓰거나 프로그램으로 읽어 들인다.
- 반환 값 : 쓰여 진 또는 읽혀 진 객체의 수를 반환. 오류가 발생하면 `count`보다 작은 값이 반환된다.

fwrite() / fread() 사용 예

➡ fwrite() / fread() 형식

```
size_t fwrite(const void *buffer, size_t size, size_t count, FILE *fp);  
size_t fread(void *buffer, size_t size, size_t count, FILE *fp);
```

➡ fwrite() / fread() 사용 예

```
FILE *fp;  
char name[10];  
int num;  
  
fwrite(name, sizeof(name), 1, fp);  
fread(name, sizeof(name), 1, fp);  
  
fwrite(&num, sizeof(int), 1, fp);  
fread(&num, sizeof(int), 1, fp);
```

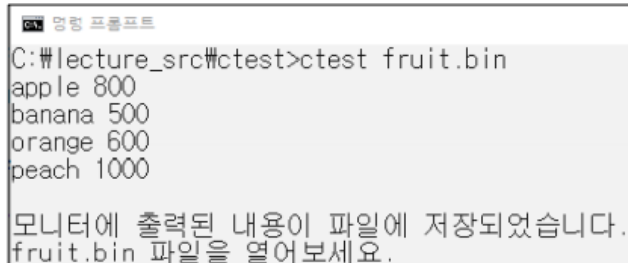
fwrite() 함수의 활용

[예제 9.4] fwrite() 함수의 활용

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4

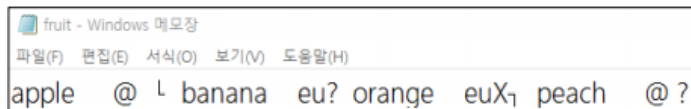
struct unit {
    char fruit[10];
    int price;
};

typedef struct unit unit;
```



```
C:\#lecture_src#ctest>ctest fruit.bin
apple 800
banana 500
orange 600
peach 1000

모니터에 출력된 내용이 파일에 저장되었습니다.
fruit.bin 파일을 열어보세요.
```



```
fruit - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
apple @ ^ banana eu? orange euX_ peach @ ?
```

```
int main(int argc, char *argv[])
{
    FILE *fp;
    unit list[SIZE] = { "apple", 800, "banana", 500, "orange", 600,
        "peach", 1000 };

    fp = fopen(argv[1], "wb"); //argv[1] = "fruit.bin"
    if (fp == NULL) {
        printf("%s Can't open file", argv[1]);
        exit(1);
    }
    for (int i = 0; i < sizeof(list) / sizeof(unit); i++) {
        fprintf(stdout, "%s %d\n", (list + i)->fruit, (list + i)->price);
    }
    fwrite(list, sizeof(list), 1, fp);

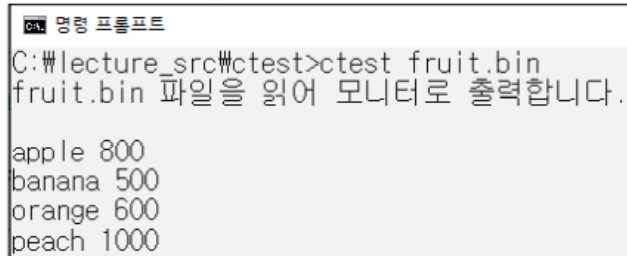
    fprintf(stdout, "\n모니터에 출력된 내용이 파일에 저장되었습니다.\n");
    fprintf(stdout, "%s 파일을 열어보세요.\n\n", argv[1]);
    fclose(fp);
    return 0;
}
```


fread() 함수의 활용

[예제 9.5] fread() 함수의 활용

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4

struct unit {
    char fruit[10];
    int price;
};
typedef struct unit unit;
```



```
C:\lecture_src\ctest>ctest fruit.bin
fruit.bin 파일을 읽어 모니터로 출력합니다.

apple 800
banana 500
orange 600
peach 1000
```

```
int main(int argc, char *argv[])
{
    FILE *fp;
    unit list[SIZE];

    fp = fopen(argv[1], "rb"); //argv[1] = "fruit.bin"
    if (fp == NULL) {
        printf("%s Can't open file", argv[1]);
        exit(1);
    }
    fprintf(stdout, "%s 파일을 읽어 모니터로 출력합니다.\n\n", argv[1]);

    fread(list, sizeof(list), 1, fp);

    for (int i = 0; i < sizeof(list) / sizeof(unit); i++) {
        fprintf(stdout, "%s %d\n", (list + i)->fruit, (list + i)->price);
    }
    fclose(fp);
    return 0;
}
```

파일의 끝과 오류 처리

a	p	p	l	e	EOF
---	---	---	---	---	-----

[그림 9.4] 파일의 끝(EOF)

➡ `feof()` 형식

```
int feof(FILE *fp);
```

- 파일의 끝이 아니면 0을 반환

➡ `feof()` 예

```
FILE *fp
```

```
char ch;
```

```
while(!feof(fp))
```

```
    fread(&ch, sizeof(char), 1, fp);
```

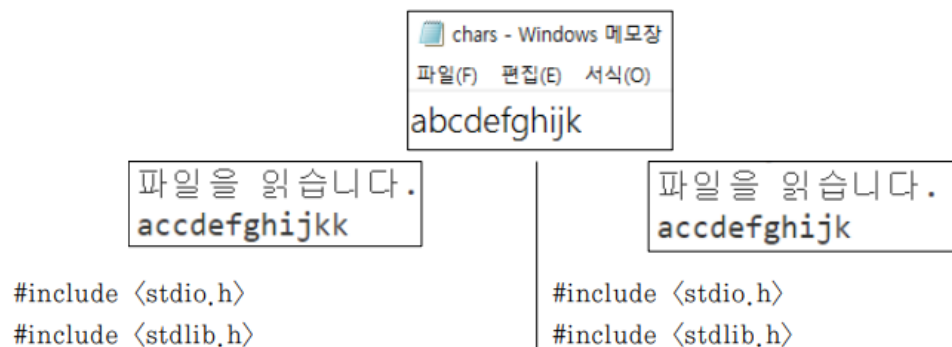
```
while(1) {
```

```
    if (feof(fp)) break; //추천하는 방법
```

```
};
```

feof() 함수의 서로 다른 사용법

[예제 9.6] feof() 함수의 서로 다른 사용법



```
int main(int argc, char *argv[])
{
    FILE *fp;
    char ch;

    fp = fopen("chars.txt", "r");
    if (fp == NULL) {
        printf("%s Can't open file", argv[1]);
        exit(1);
    }

    printf("파일을 읽습니다.\n");

    while (!feof(fp)) {
        fscanf(fp, "%c", &ch);
        printf("%c", ch);
    };
    fclose(fp);
    return 0;
}
```

```
int main(int argc, char *argv[])
{
    FILE *fp;
    char ch;

    fp = fopen("chars.txt", "r");
    if (fp == NULL) {
        printf("%s Can't open file", argv[1]);
        exit(1);
    }

    printf("파일을 읽습니다.\n");

    while(1) {
        fscanf(fp, "%c", &ch);
        if (feof(fp)) break;
        printf("%c", ch);
    };
    fclose(fp);
    return 0;
}
```

파일의 끝과 오류 여부 판단

➡ `ferror()` 형식

```
int ferror(FILE *fp);
```

- 오류가 발생하지 않았으면 0을 리턴

➡ `ferror()` 예

```
fread(&ch, sizeof(char), 1, fp);  
if(ferror(fp))  
    printf("file error");
```

[예제 9.7] 파일의 끝과 오류 여부 판단

```
#include <stdio.h>  
#include <stdlib.h>  
  
int main(int argc, char *argv[])  
{  
    FILE *fp;  
    char ch;
```

```
C:\lecture_src\ctest>ctest fruit.txt  
apple 800  
banana 500  
orange 600  
peach 1000
```

```
fp = fopen(argv[1], "r"); //argv[1]="fruit.txt"  
if (fp == NULL) {  
    printf("%s Can't open file", argv[1]);  
    exit(1);  
}
```

```
while (1) {  
    fread(&ch, sizeof(char), 1, fp);  
  
    if (ferror(fp)) {  
        printf("file read error");  
        break;  
    }  
    if (feof(fp)) break;  
  
    fprintf(stdout, "%c", ch);  
}  
fclose(fp);  
return 0;  
}
```

랜덤 파일 처리

□ fseek() 형식

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

- 이 함수가 성공적으로 이동한 경우에는 0 을 반환.
- 오프셋 바이트의 길이는 long 형으로 뒤에 'L'를 붙임.

□ fseek() 예

```
fseek(fp, 0L, SEEK_SET); //파일 포인터 fp를 파일의 처음으로 이동
fseek(fp, -9L, SEEK_SET); //오류
fseek(fp, 9L, 0);          //fp를 파일의 처음에서 9bytes 뒤로 이동
fseek(fp, -9L, SEEK_END); //fp를 파일의 끝에서 9bytes 앞으로 이동
fseek(fp, 9L, SEEK_CUR);  //fp를 현재 위치에서 9bytes 뒤로 이동
fseek(fp, -9L, 1);        //fp를 현재 위치에서 9bytes 앞으로 이동
fseek(fp, 0L, 2);         //fp를 파일의 끝으로 이동
fseek(fp, 0L, SEEK_END);  //fp를 파일의 끝으로 이동
```

[표 9.2] 기준점과 오프셋 범위

기준점		의미	오프셋 범위
SEEK_SET	0	파일의 처음	0 ~ 양수
SEEK_CUR	1	현재 위치	음수 ~ 양수
SEEK_END	2	파일의 끝	음수 ~ 0

⇒ ftell() 형식

```
long ftell(FILE *fp);
```

- 이 함수가 성공적으로 수행되면 현재의 파일 포인터 위치를 반환.
- 오류가 발생하면 -1을 반환.

⇒ fseek() 예

```
FILE *fp;
int position;
position = ftell(fp);
```

fseek()과 ftell() 함수의 활용

[예제 9.8] fseek()과 ftell() 함수의 활용

```
#include <stdio.h>
#include <stdlib.h>
#define POS 2

int main(int argc, char *argv[])
{
    FILE *fp;
    int nums[9] = {9, 8, 7, 6, 5, 4, 3, 2, 1};
    int position, last2second;

    fp = fopen(argv[1], "wb"); //argv[1]="nums.data"
    if (fp == NULL) {
        printf("%s Can't open file", argv[1]);
        exit(1);
    }
    fwrite(nums, sizeof(nums), 1, fp); //nums 배열을 파일에 쓴다.
    fclose(fp);
    printf("데이터를 %s에 저장했습니다.\n", argv[1]);
}
```

명령 프롬프트

```
C:\lecture_src\ctest>ctest nums.data
데이터를 nums.data에 저장했습니다.
fp의 처음 위치 = 0
fp의 마지막에서 두 번째 정수의 위치 = 28
fp의 마지막에서 두 번째 정수의 값 = 2
```

```
fp = fopen(argv[1], "rb"); //argv[1]="nums.data"
if (fp == NULL) {
    printf("%s Can't open file", argv[1]);
    exit(1);
}
```

```
position = ftell(fp); //파일 포인터의 위치를 읽음
printf("fp의 처음 위치 = %d\n", position);
```

```
fseek(fp, -(sizeof(int)*POS), SEEK_END); //파일 포인터 이동
position = ftell(fp);
printf("fp의 마지막에서 두 번째 정수의 위치 = %d\n", position);
```

```
fread(&last2second, sizeof(int), 1, fp);
printf("fp의 마지막에서 두 번째 정수의 값 = %d\n", last2second);
fclose(fp);
return 0;
}
```

기타 파일 입출력 함수들

□ 형식

`int fgetc(FILE *fp)`

- `fp` 와 연결된 파일에서 하나의 문자를 읽음.

`int fputc(int ch, FILE *fp)`

- `fp` 와 연결된 파일에 하나의 문자를 출력.

`char *fgets(char *str, int maxcount, FILE *fp)`

- `fp` 와 연결된 파일에서 `maxcount` 만큼의 문자(문자열)를 읽거나
- 줄 바꿈 문자가 있을 때까지 읽음.
- 문자열의 끝에는 `NULL('\0')` 문자를 자동으로 추가.

`int fputs(const char *str, FILE *fp)`

- `fp` 와 연결된 파일에 `str` 에 저장된 문자열을 출력.

두 텍스트 파일의 내용 비교

[예제 9.9] 두 텍스트 파일의 내용 비교

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

```
int main(int argc, char *argv[])
{
```

```
    FILE *fpa, *fpb;
    char cha, chb;
```

```
    fpa = fopen(argv[1], "r");
    if (fpa == NULL) {
        printf("%s Can't open file", argv[1]);
        exit(1);
    }
```

```
    fpb = fopen(argv[2], "r");
    if (fpb == NULL) {
        printf("%s Can't open file", argv[2]);
        exit(1);
    }
```

명령 프롬프트

C:\lecture_src\ctest>ctest fruit.txt fruit.txt
fruit.txt와 fruit.txt의 내용은 같습니다.

C:\lecture_src\ctest>ctest fruit.txt chars.txt
fruit.txt와 chars.txt의 내용은 다릅니다.

```
        while(1) {
            cha = fgetc(fpa);
            chb = fgetc(fpb);

            if(cha != chb) {
                printf("%s와 %s의 내용은 다릅니다.\n", argv[1], argv[2]);
                break;
            } else if((cha==EOF) && (chb==EOF)) {
                printf("%s와 %s의 내용은 같습니다.\n", argv[1], argv[2]);
                break;
            }

            fclose(fpa);
            fclose(fpb);
            return 0;
        }
```


파일 입출력 응용 문제

[예제 9.10] 파일 입출력 응용문제

아래의 형식으로 저장된 파일이 sales.txt의 내용입니다. 파일을 읽어 총 판매 금액과 총 판매금액 대비 제품 'AA' 판매금액의 비율을 제시된 출력 형태와 같이 표현되도록 프로그램을 작성하세요.

제품 번호	제품 구분	거래처 번호	단가	판매량
숫자	문자	숫자	숫자	숫자
8자리	2자리	6자리	5자리	3자리

sales.txt

```
91010101 AA 111111 01000 100
91010201 BB 222222 01500 150
91010301 CC 333333 02000 200
91010402 AA 333333 01000 250
91010502 BB 111111 03000 100
91020102 CC 222222 00500 350
91020203 AA 111111 01000 150
91020303 BB 333333 01500 200
91020403 CC 333333 02000 250
91020504 AA 111111 01000 100
91030104 BB 111111 03000 150
```

[예제 9.11] 파일 입출력 응용문제 해답

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

struct unit {
    int goods_num; //제품번호, 8자리
    char goods[3]; //제품구분, 2자리
    int client; //거래처 번호, 6자리
    int price; //단가, 5자리
    int sales; //판매량, 3자리
};

typedef struct unit unit;
```

```
명령 프롬프트
C:\lecture_src\ctest>ctest sales.txt
91010101 AA 111111 1000 100
91010201 BB 222222 1500 150
91010301 CC 333333 2000 200
91010402 AA 333333 1000 250
91010502 BB 111111 3000 100
91020102 CC 222222 500 350
91020203 AA 111111 1000 150
91020303 BB 333333 1500 200
91020403 CC 333333 2000 250
91020504 AA 111111 1000 100
91030104 BB 111111 3000 150
총 판매금액 = 2950000
AA제품의 총 판매 금액 = 2350000
총 판매금액 대비 제품 AA 판매금액 비율 = 79.66%
```

파일 입출력 응용문제 해답 (1)

```
int read_file(unit *datalist, char *filename)
{
    FILE *fp;
    int i = 0;

    fp = fopen(filename, "r"); //filename = "sales.txt"
    if (fp == NULL) {
        printf("%s Can't open file", filename);
        exit(1);
    }
    while(1) {
        fscanf(fp, "%8d", &(datalist + i)->goods_num);
        fscanf(fp, "%s", (datalist + i)->goods);
        fscanf(fp, "%6d", &(datalist + i)->client);
        fscanf(fp, "%5d", &(datalist + i)->price);
        fscanf(fp, "%3d", &(datalist + i)->sales);

        if (feof(fp)) break;
        i++;
    }
    fclose(fp);
    return i;
}
```

```
void print_datalist(unit *datalist, int len) {
    for (int i = 0; i < len; i++) {
        printf("%8d ", (datalist + i)->goods_num);
        printf("%s ", (datalist + i)->goods);
        printf("%6d ", (datalist + i)->client);
        printf("%5d ", (datalist + i)->price);
        printf("%3d\n", (datalist + i)->sales);
    }
}

int total_amount(unit *datalist, int len) {
    int total = 0;

    for (int i = 0; i < len; i++) {
        total += ( (datalist + i)->price *(datalist + i)->sales );
    }
    return total;
}
```

파일 입출력 응용문제 해답 (2)

```
int goods_amount(unit *datalist, int len, char *goods) {
    int total = 0;

    for (int i = 0; i < len; i++) {
        if (strcmp((datalist + i)->goods, goods)) {
            total += ((datalist + i)->price * (datalist + i)->sales);
        }
    }
    return total;
}
```

```
int main(int argc, char *argv[])
{
    unit data[20];
    int len, total, total_AA;

    len = read_file(data, argv[1]);
    print_datalist(data, len);

    total = total_amount(data, len);
    printf("총 판매금액 = %d\n", total);

    total_AA = goods_amount(data, len, "AA");
    printf("AA제품의 총 판매 금액 = %d\n", total_AA);

    printf("총 판매금액 대비 제품 AA 판매금액 비율 = %.2f%%\n", (total_AA
/(double)total) * 100);
    return 0;
}
```

개념 확인학습 & 적용 확인학습 & 응용 프로그래밍

- 다음 파일에 있는 문제들의 해답을 스스로 작성 해 보신 후 개념 & 적용 확인 학습 영상을 학습 하시기 바랍니다.
 - c_09장_파일입출력_ex.pdf
- 퀴즈와 과제가 출제되었습니다.
 - 영상을 학습하신 후 과제와 퀴즈를 수행 하시기 바랍니다.

Q & A

- “포인터”에 대한 학습이 모두 끝났습니다.
- 모든 내용을 이해 하셨나요?
- 아직 이해가 안되는 내용이 있다면 다시 한번 복습하시기 바랍니다.
- 질문은 한림 SmartLEAD 쪽지 또는 e-mail 또는 전화상담을 이용하시기 바랍니다.



- 퀴즈와 과제가 출제되었습니다. 마감시간에 늦지 않도록 주의해 주세요.
- 다음 시간에는 “파일입출력”에 대해 알아보겠습니다.
- 수고하셨습니다.^^