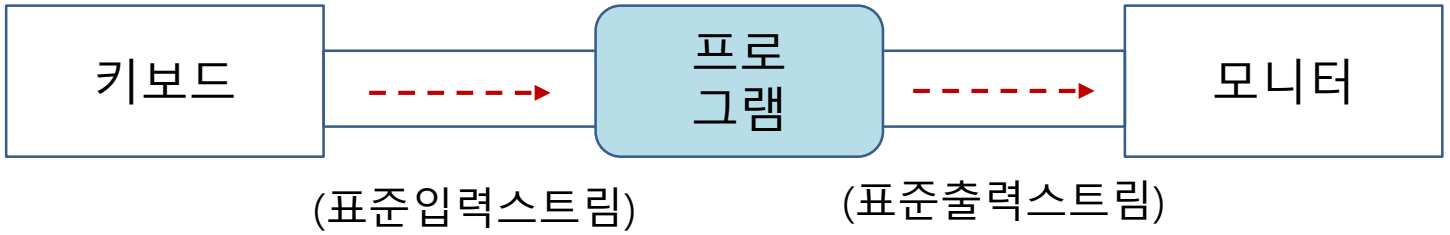
C – 파일 입출력

file IO

스트림(stream)

- **스트림(Stream)**

  스트림이란 입출력 장치와 데이터(파일, 프로그램 등)를 연결시켜주는 통로(다리) 역할을 한다.



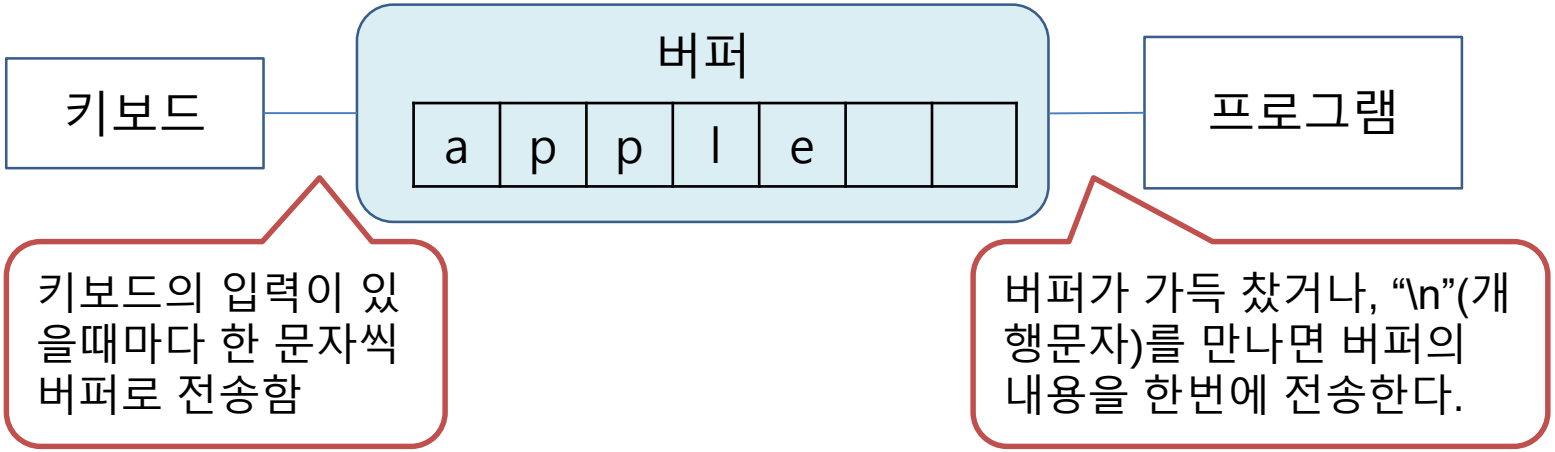
스트림	설명	장치
stdin	표준 입력을 담당	키보드로 입력
stdout	표준 출력을 담당	모니터로 출력
stderr	표준 에러를 담당	모니터로 출력

버퍼(Buffer)

- **버퍼(Buffer)**

  스트림은 내부에 문자 배열 형태의 버퍼라는 임시 메모리 공간을 가지고 있다. 입력 버퍼는 데이터를 저장하기 위한 버퍼이며, 출력 버퍼는 데이터를 출력하기 위한 버퍼이다.

(입력 스트림)



표준 입출력

- 입력 버퍼의 동작

  입력 버퍼의 문제는 문자 형식을 입력 받을때 발생한다. 엔터를 칠때 남아 있는 문자 '\n' 때문이다.

scanf()로 학번 입력

1	0	1	'\n'		

```
while (getchar() != '₩n');
```

fgets()는 '\n'을 가져가버려 자동으로 출력된다.

```
학번 입력: 101
이름 입력: 학번: 101
이름:
```

▶ 해결 방법

while (getchar() != '\n');

표준 입출력

- 입력 버퍼의 동작

```c
int no;  //학번
char name[10]; //이름

printf("학번 입력: ");
scanf("%d", &no);

//'\n' 앞까지 버퍼를 비운다.
while (getchar() != '\n');

printf("이름 입력: ");
//scanf("%s", name);
fgets(name, sizeof(name), stdin);

printf("학번: %d\n", no);
printf("이름: %s\n", name);
```

```
학번 입력: 101
이름 입력: 신유빈
학번: 101
이름: 신유빈
```

파일 입출력 관련 함수

- 파일 입출력 함수 - 헤더파일 <stdio.h> 포함

함수의 원형	기능 설명
fputc(int c, FILE* stream**)**	파일에 문자 1개 쓰기
fputs(const char* s, **FILE*** stream**)**	파일에 문자열 쓰기
fgetc(FILE* stream**)**	파일에서 한 문자 읽기, 파일의 끝에 도달
fgets(char* s, int MaxCount, **FILE*** stream**)**	파일로부터 문자열 입력 받음
fprintf(FILE* stream, const char* format**)**	파일로부터 자료형에 맞춰 데이터를 입력
fscanf_s(FILE* stream, const char* format, ...**)**	파일에 자료형에 맞춰 데이터를 쓰기

파일 입출력

- 파일 입출력의 필요성

  프로그램 실행 중에 메모리에 저장된 데이터는 프로그램이 종료되면 사라진다.

  데이터를 프로그램이 종료된 후에도 계속해서 사용하려면 파일에 저장하고

  필요할때 파일을 읽어서 데이터를 사용할 수 있다.

- 파일 입출력의 과정(process)

1. 파일 스트림을 생성한다 -> 파일 포인터 생성(**FILE* fp**)
2. 파일을 연다. -> **fopen()** 함수
3. 파일 입출력을 수행한다. -> fgetc(), fputc(), fgets(), fputs() 등
4. 파일을 닫는다. ->**fclose()**

파일 입출력

- 파일 입출력 함수

함수의 원형	모드 구분	기능 설명
fopen(const char* filename, const char* mode**)**	**fopen(파일, “w”)**	파일에 쓴다.
	fopen(파일, “r”)	파일을 읽는다.
	fopen(파일, “a”)	파일에 추가로 쓴다.
fclose(FILE* stream**)**		파일을 닫음

파일 입출력

- 파일 쓰기

```c
#define _CRT_SECURE_NO_WARNINGS //fopen() 처리
#include <stdio.h>

int main()
{
    FILE* fp; //파일 구조체 포인터 변수 선언

    fp = fopen("c:/cfile/out.txt", "w");  //절대 경로

    if (fp == NULL) {
        printf("파일 열기에 실패함\n");
        return 1;
    }
```

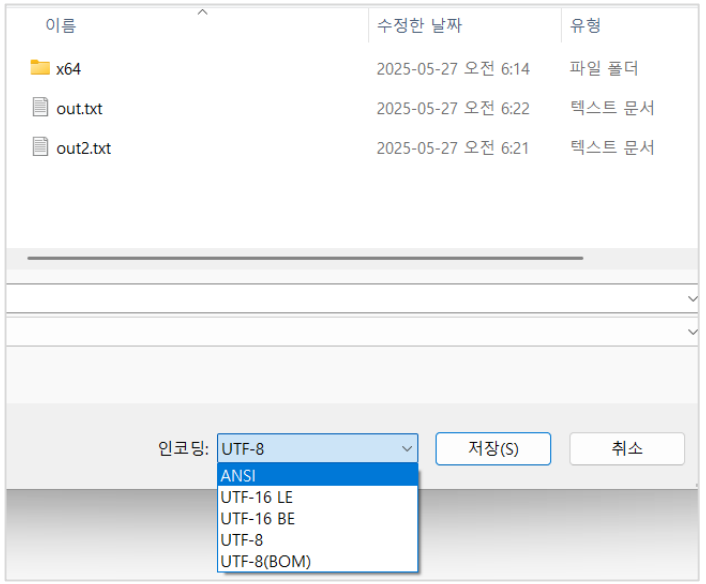
out.txt

```
Hello
Apple

사과
```

파일 입출력

- 파일 쓰기 – fputs()

```
    //문자 1개 쓰기
    fputc('H', fp);
    fputc('e', fp);
    fputc('l', fp);
    fputc('l', fp);
    fputc('o', fp);

    //문자열 쓰기
    fputs("\nApple\n", fp);  //영어
    fputs("\n좋아요\n", fp); //한글

    fclose(fp); //파일 닫기

    printf("파일 쓰기 완료!");
    return 0;
}
```



@ 텍스트 파일에서 한글이 깨지는 경우

메모장을 다른 이름으로 저장하여
인코딩 구분을 ANSI 모드로 변경함

파일 입출력

- 파일 읽기

```
FILE* fp; //파일 포인터 변수
int ch;  //읽은 문자 변수(코드값이므로 int형)

fp = fopen("c:/cfile/out.txt", "r"); //읽기 모드 - "r"
if (fp == NULL) {
    printf("파일 열기에 실패함\n");
    return 1;  //에러시 1 or -1
}

//문자 1개 읽기
/*ch = fgetc(fp);
printf("%c", ch);*/ //'H'
```

파일 입출력

- 파일 읽기 – fgetc()

```c
//모든 글자 읽기
//방법 1
/*while (1) {
    ch = fgetc(fp);
    if (ch == EOF) break;
    //if ((ch = fgetc(fp)) == EOF) break;
    printf("%c", ch);
}*/

//방법 2
while ((ch = fgetc(fp)) != EOF) {
    printf("%c", ch);
}

fclose(fp);
```

```
Hello
Apple

좋아요
```

파일 입출력

- 아스키 파일 쓰기

```c
FILE* fp; //파일 포인터 변수
int i;

//fopen_s(파일포인터, 파일이름, 쓰기모드)
//상대 경로(프로젝트 폴더 안에 저장됨)
fopen_s(&fp, "ascii.txt", "w");
if (fp == NULL) {
    printf("파일 열기에 실패함\n");
    return 1;  //에러시 1 or -1
}

printf("========== ASCII 테이블 ==========\n");
for (i = 32; i < 128; i++) { //32번 공백문자
    if (i % 10 == 0)
        fputc('\n', fp); //줄바꿈
    fputc(i, fp);
    fputc('\t', fp);
}

fclose(fp);
```

ascii.txt

```
    !   "   #   $   %   &   '
(   )   *   +   ,   -   .   /   0   1
2   3   4   5   6   7   8   9   :   ;
<   =   >   ?   @   A   B   C   D   E
F   G   H   I   J   K   L   M   N   O
P   Q   R   S   T   U   V   W   X   Y
Z   [   ₩   ]   ^   _   `   a   b   c
d   e   f   g   h   i   j   k   l   m
n   o   p   q   r   s   t   u   v   w
x   y   z   {   |   }   ~   □
```

파일 입출력

- 아스키 파일 읽기

```
FILE* fp; //파일 포인터 변수
int ch;   //아스키 코드를 저장할 변수

fopen_s(&fp, "ascii.txt", "r");
if (fp == NULL) {
    printf("파일 열기에 실패함\n");
    return 1;  //에러시 1 or -1
}

//파일 읽기
while ((ch = fgetc(fp)) != EOF) {
    printf("%c", ch);
}

fcolse(fp);
```

파일 입출력

- 파일 쓰기(추가 저장)

```c
FILE* fp;
char msg[] = "행운을 빌어요~";

fp = fopen("c:/cfile/out.txt", "a");  //추가 모드 - "a"
if (fp == NULL) {
    printf("파일 열기에 실패함\n");
    return 1;
}

//문자열 쓰기
fputs("Good Luck~\n", fp);
fprintf(fp, "%s\n", msg); //서식 문자 사용

fclose(fp);

printf("파일 추가 쓰기 완료!");
```

out.txt

```
Hello
Apple

사과
Good Luck~
행운을 빌어요~
```

파일 입출력

- 문자열 쓰고 읽기

```
//파일 쓰기
FILE* fp;
char str[] = "abcdefg\nhijklmn\nopqrstu\nvwxyz";

fp = fopen("data.txt", "w");  //상대 경로
if (fp == NULL)
    return 1;

fprintf(fp, "%s", str);

fclose(fp);
```

파일 입출력

- 문자열 쓰고 읽기

```c
//파일 읽기
char buf[256];
int i = 1;

fp = fopen("data.txt", "r");
if (fp == NULL)
    return 1;

//파일의 끝까지 읽기
while (fgets(buf, sizeof(buf), fp) != NULL) {
    printf("%03d: %s", i, buf);
    i++;
}
fclose(fp);
```

```
001: abcdefg
002: hijklmn
003: opqrstu
004: vwxyz
```

영어 단어 쓰고 읽기

- 영어 단어 쓰고 읽기

```
void wordWrite();
void wordRead();
int main()
{
    //영어 단어 쓰기
    wordWrite();

    //영어 단어 읽기
    wordRead();

    return 0;
}
```

word.txt

ant bear chicken cow dog elephant monkey lion tiger horse snake

```
ant bear chicken cow dog elephant monkey lion tiger horse snake
```

영어 단어 쓰기

- 영어 단어 쓰기 – 포인터 배열

```
void wordWrite()
{
    FILE* fp; //파일 포인터 변수

    if (fopen_s(&fp, "word.txt", "w") != 0) {
        perror("파일 열기에 실패함\n");
        return 1;  //에러시 1 or -1
    }

    char* words[] = { "ant", "bear", "chicken", "cow", "dog", "elephant",
        "monkey", "lion", "tiger", "horse", "snake" };

    int wordCount = sizeof(words) / sizeof(words[0]);
    //printf("%d\n", wordCount);

    for (int i = 0; i < wordCount; i++) {
        fprintf(fp, "%s ", words[i]);
    }

    fclose(fp);
}
```

영어 단어 읽기

- 영어 단어 읽기 – 포인터 배열

```
void wordRead()
{
    FILE* fp; //파일 포인터 변수
    char str[256]; //읽은 문자열을 저장할 배열 선언
    char* wordList[MAX_WORDS]; //분리된 단어를 저장할 배열
    int idxOfWordList = 0;

    if (fopen_s(&fp, "word.txt", "r") != 0) {
        perror("파일 열기에 실패함\n");
        return 1;  //에러시 1 or -1
    }

    //파일 읽기
    printf("***** 읽은 내용 *****\n");
    while (fgets(str, sizeof(str), fp) != NULL) {
        printf("%s", str);
    }
    printf("\n");
```

영어 단어 읽기

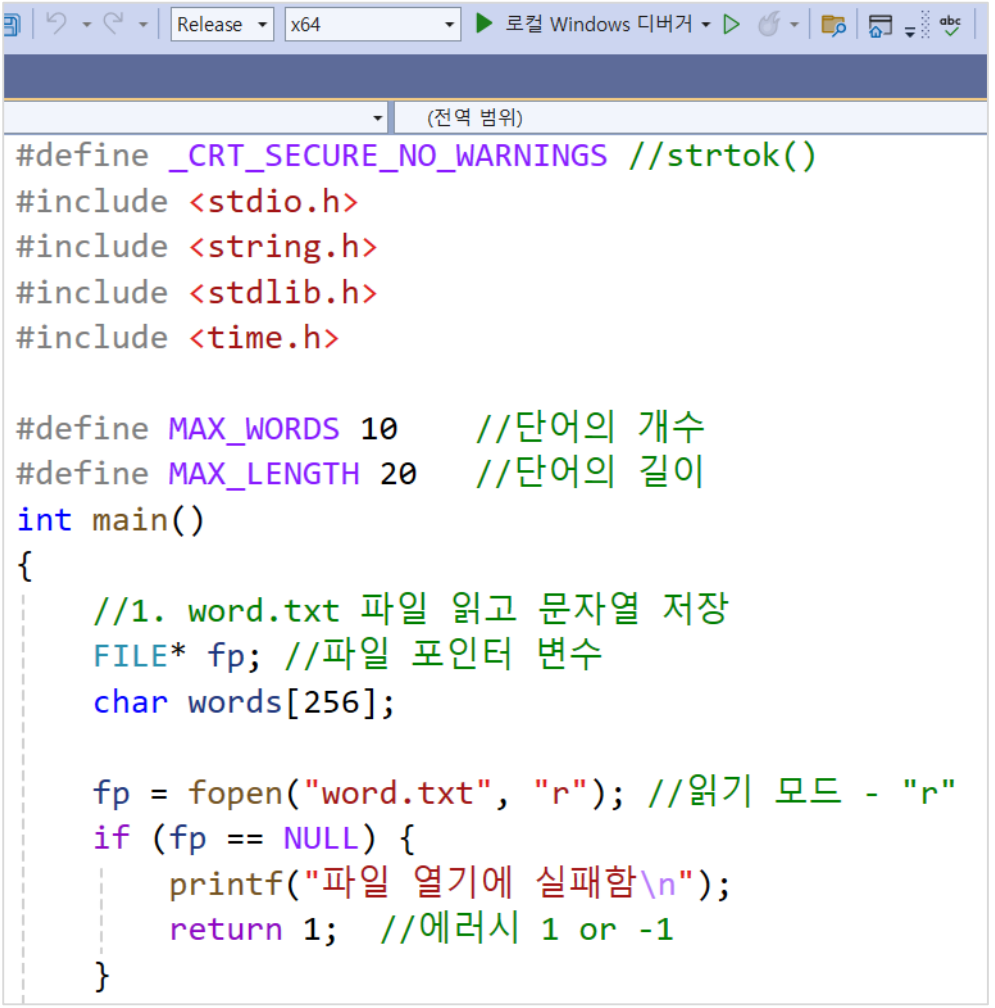
- 영어 단어 읽기 – 포인터 배열

```
    //랜덤 추출
    printf("***** 단어 추출(랜덤) *****\n");
    char* word = strtok(str, " ");
    //printf("첫번째 단어: %s\n", ptr);

    while (word != NULL && idxOfWordList < MAX_WORDS ) {
        wordList[idxOfWordList++] = word;
        word = strtok(NULL, " ");
    }
    //printf("%d\n", idxOfWordList);
    //printf("마지막 단어: %s\n", wordList[--idxOfWordList]);

    srand(time(NULL));

    int randIdx = rand() % idxOfWordList;
    printf("선택 단어: %s\n", wordList[randIdx]);

    fclose(fp);
}
```

```
***** 읽은 내용 *****
ant bear chicken cow dog elephant monkey lion tiger horse snake
***** 단어 추출(랜덤) *****
선택 단어: chicken
```

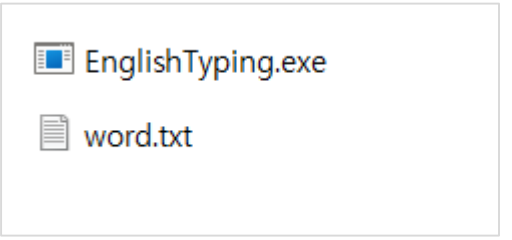
영어 타자 게임

- 프로젝트 배포 – Release 모드




```c
#define _CRT_SECURE_NO_WARNINGS //strtok()
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

#define MAX_WORDS 10    //단어의 개수
#define MAX_LENGTH 20   //단어의 길이
int main()
{
    //1. word.txt 파일 읽고 문자열 저장
    FILE* fp; //파일 포인터 변수
    char words[256];

    fp = fopen("word.txt", "r"); //읽기 모드 - "r"
    if (fp == NULL) {
        printf("파일 열기에 실패함\n");
        return 1;  //에러시 1 or -1
    }
```

typing_game 디렉터리




```
[영어 타자 게임], 준비되면 엔터>

문제 1
ant
ant
통과!

문제 2
monkey
monkey
통과!

문제 3
dog
dog
통과!

문제 4
lion
lion
통과!
```

영어 타자 게임

```c
//문자열 가져오기
while (fgets(words, sizeof(words), fp) != NULL) {
    //printf("%s", words);
}

// 2. 단어 분리 및 저장
char* wordList[MAX_WORDS]; // 분리된 단어 저장용 배열
int idxOfWords = 0;

char* ptr = strtok(words, " "); //공백을 구분기호로 문자열 자르기
while (ptr != NULL && idxOfWords < MAX_WORDS) {
    wordList[idxOfWords++] = ptr;
    ptr = strtok(NULL, " ");
}

// 2. 타자 게임 준비
char* question;
char* answer = (char*)malloc(MAX_LENGTH * sizeof(char)); //동적 배열로 할당
int n = 1;
clock_t start, end;
double elapsedTime;
srand(time(NULL));
```

영어 타자 게임

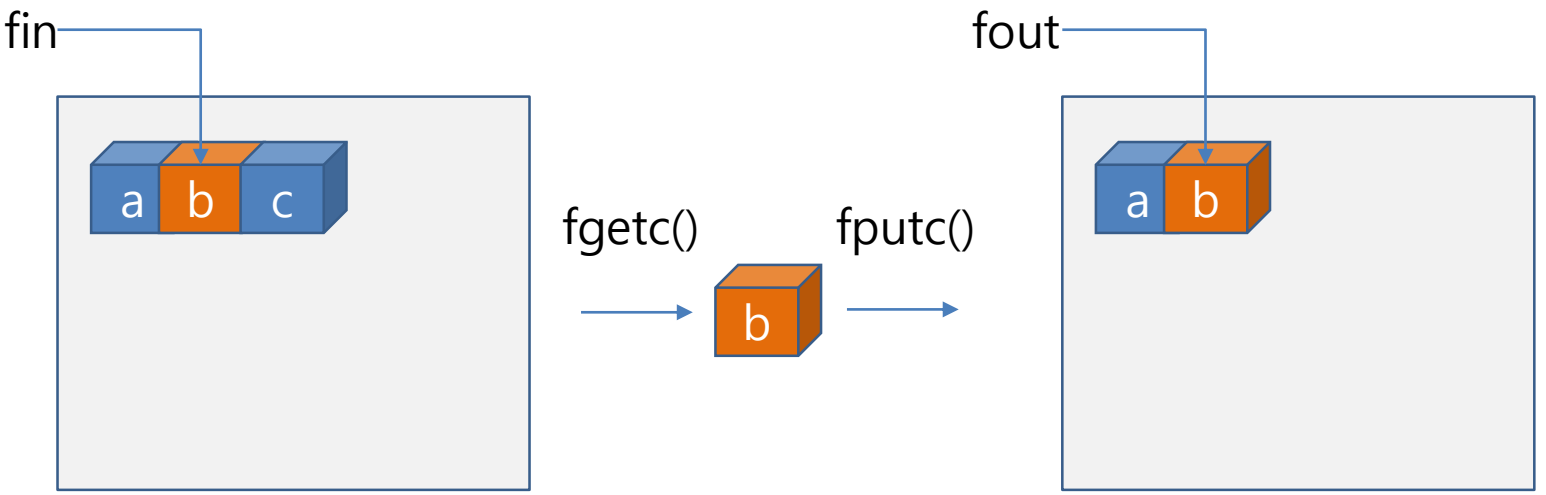
```
printf("[영어 타자 게임], 준비되면 엔터> ");
getchar();
start = clock();
while (n <= 10) {
    printf("\n문제 %d\n", n);
    int rndIdx = rand() % idxOfWords; //실제 단어 개수 사용
    question = wordList[rndIdx];  //랜덤한 단어 추출
    printf("%s\n", question); //문제 출제
    scanf("%s", answer);  //사용자 입력
    if (strcmp(question, answer) == 0) {
        printf("통과!\n");
        n++;
    }else {
        printf("오타! 다시 도전!\n");
    }
}
end = clock();
elapsedTime = (double)(end - start) / CLOCKS_PER_SEC;
printf("게임 소요 시간: %.2f초\n", elapsedTime);
free(answer); // 메모리 해제

system("pause");
```

파일 복사

- 파일 복사 – 파일 읽고 쓰기

fopen_s(&fin, 원본파일, "r")

fopen_s(&fout, 복사파일, "w")

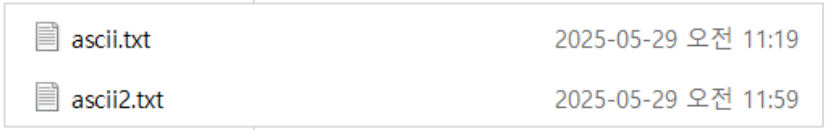


fin은 원본파일을 가리키는 파일 포인터이며, fout은 새로 생성할 파일을 가리키는 파일 포인터로 두 개의 파일을 오픈하게 된다.

파일 복사

- 파일 복사 – 읽고 쓰기

```c
FILE* fin;     //읽기 파일 포인터 변수
FILE* fout;    //쓰기 파일 포인터 변수
int input = 0; //문자 코드값 저장

fopen_s(&fin, "ascii.txt", "r");
fopen_s(&fout, "ascii2.txt", "w");

if (fin == NULL || fout == NULL) {
    printf("파일 열기에 실패함\n");
    return 1;
}

puts("===== 파일에 데이터 쓰기(저장) =====\n");
while (input != EOF) {
    input = fgetc(fin);  //문자 코드값 저장
    fputc(input, fout);  //파일에 쓰기
}

fclose(fin);
fclose(fout);
```



실습 문제 – 파일에 구구단 쓰기

아래의 코드 작성 부분을 완성하여 구구단을 파일에 저장하세요.

```c
FILE* fp;

fopen_s(&fp, "gugudan.txt", "w");
if (fp == NULL) {
    perror("파일 열기에 실패함\n");
    return 1;
}

//코드 작성



fclose(fp);
```

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18

3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
```

```
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72

9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
```

성적 관리

- 입력 받아 저장하기 – 추가 쓰기

```c
FILE* fp;
char name[20];
int eng, math;

//표준 입력 스트림 지정(stdin)
printf("이름 입력: ");
//scanf("%s", name);
fscanf(stdin, "%s", name);

printf("영어점수 입력: ");
//scanf("%d", &eng);
fscanf(stdin, "%d", &eng);

printf("수학점수 입력: ");
//scanf("%d", &math);
fscanf(stdin, "%d", &math);
```

```
이름 입력: 한강
영어점수 입력: 95
수학점수 입력: 87
한강 95 87
```

성적 관리

- 입력 받아 저장하기 – 추가 쓰기

```
//파일 열기(추가 쓰기 모드)
fopen_s(&fp, "score.txt", "a");
if (fp == NULL) {
    printf("파일 열기에 실패함\n");
    return -1;
}

//파일에 쓰기
fprintf(fp, "%s %d %d\n", name, eng, math);

//표준 출력 스트림 지정(stdout)
printf("%s %d %d\n", name, eng, math);
//fprintf(stdout, "%s %d %d\n", name, eng, math);

fclose(fp);
```

성적 관리

- score 파일 읽기

```c
FILE* fp;
char name[20];
int eng, math;
char line[256]; //한 줄을 저장할 버퍼

fopen_s(&fp, "score.txt", "r");
if (fp == NULL) {
    printf("파일 열기에 실패함\n");
    return -1;
}

//파일을 읽은 후 내용 출력
printf("이름 영어 수학\n");
while (fgets(line, sizeof(line), fp) != NULL) {
    fprintf(stdout, "%s", line);
}

fclose(fp);
```

```
이름 영어 수학
한강 95 87
임서현 80 91
```

성적 관리

- 성적표 만들기

```
번호 입력(0이하 종료): 1
이름 입력: 이우주
영어점수 입력: 80
수학점수 입력: 90
번호 입력(0이하 종료): 2
이름 입력: 정은하
영어점수 입력: 90
수학점수 입력: 85
번호 입력(0이하 종료): 3
이름 입력: 강하늘
영어점수 입력: 70
수학점수 입력: 75
번호 입력(0이하 종료): 0


===== 성  적  표 =====
번호 이름  영어  수학
  1  이우주  80  90
  2  정은하  90  85
  3  강하늘  70  75
```

scorelist.txt

```
번호 이름 영어 수학
 1 이우주 80 90
 2 정은하 90 85
 3 강하늘 70 75
```

성적표 만들기

```
FILE* fout;
char name[20];
int no, eng, math; //번호, 수학, 영어점수

fopen_s(&fout, "scorelist.txt", "w");
if (fout == NULL) {
    printf("파일 열기에 실패함\n");
    return -1;
}

fprintf(fout, "번호 이름  영어  수학\n");  //제목행
while (1) {
    printf("번호 입력(0이하 종료): ");
    scanf("%d", &no);
    if (no <= 0) break;

    printf("이름 입력: ");
    scanf("%s", name);

    printf("영어점수 입력: ");
    scanf("%d", &eng);
```

성적표 만들기

```
    printf("수학점수 입력: ");
    scanf("%d", &math);

    //파일에 쓰기
    fprintf(fout, "%3d %7s %3d %3d\n", no, name, eng, math);
}
fclose(fout);

//scorelist.txt 읽기
FILE* fin;
char line[256];

fopen_s(&fin, "scorelist.txt", "r");
if (fin == NULL) {
    printf("파일 열기에 실패함\n");
    return -1;
}
printf("\n\n===== 성  적  표 =====\n");
while (fgets(line, sizeof(line), fin) != NULL) {
    fprintf(stdout, "%s", line);
}
fclose(fin);
```

성적 관리

- 성적 리스트에서 평균 계산하기

```
FILE *fin, *fout;
char line[256];  //읽은 내용 저장
char name[20];   //이름
int no, eng, math; //번호, 수학, 영어점수
float avg;  //평균

//scorelist.txt 읽기
if(fopen_s(&fin, "scorelist.txt", "r") != 0){
    perror("파일 열기에 실패함\n");
    return -1;
}

//scorelist2.txt에 쓰기
if (fopen_s(&fout, "scorelist2.txt", "w") != 0) {
    perror("파일 열기에 실패함\n");
    return -1;
}
```

```
번호 이름   영어   수학   평균
 1  이우주   80     90   85.00
 2  정은하   90     85   87.50
 3  강하늘   70     75   72.50
```

성적 관리

- 성적 리스트에서 평균 계산하기

```
fprintf(fout, "번호 이름  영어  수학  평균\n");  //제목행
fgets(line, sizeof(line), fin); //제목행 읽음(계산에서 제외)

while (fgets(line, sizeof(line), fin) != NULL ){
    //sscanf()는 성공적으로 추출한 항목의 개수를 반환함
    if (sscanf(line, "%d %s %d %d", &no,
                name, &eng, &math) == 4) {
        //평균 계산
        avg = (float)(eng + math) / 2;
        fprintf(fout, "%2d %7s %3d %5d %6.2f\n", no, name, eng, math, avg);
    }
    else {
        printf("잘못된 형식의 데이터: %s", line);
    }
}
```

성적 관리

- 성적 리스트에서 평균 계산하기

```
//scorelist2.txt 읽기
FILE* fp;
char str[256];

if (fopen_s(&fp, "scorelist2.txt", "r") != 0) {
    perror("파일 열기에 실패함\n");
    return -1;
}

while (fgets(str, sizeof(str), fp) != NULL) {
    fprintf(stdout, "%s", str);
}

fclose(fp);
```

바이너리 파일 입출력

- 바이너리 파일 입출력 함수

모드 구분	기능 설명
fopen(파일, “wb”)	바이너리 파일에 쓴다.
fopen(파일, “rb”)	바이너리 파일을 읽는다.
fopen(파일, “ab”)	바이너리 파일에 추가로 쓴다.

함수의 원형	기능 설명
fread(void* buffer, ElementSize, ElementCount, FILE* stream**)**	바이너리 파일 읽기
fwrite(void* buffer, **ElementSize, ElementCount, FILE*** stream**)**	바이너리 파일 쓰기

바이너리 파일 입출력

- 바이너리 파일 입출력

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

int main() {
    //바이너리 파일 쓰고 읽기
    int buf1[4] = { 0xff, 0x56, 0x78, 0xfa };
    int buf2[4];

    FILE* fp;

    fp = fopen("data.dat", "wb");
    if (fp == NULL) {
        printf("파일 열기에 실패함\n");
        return 1;
    }

    //쓰기
    fwrite(buf1, sizeof(int), 4, fp);

    fclose(fp);  //파일 종료
```

바이너리 파일 입출력

- 바이너리 파일 입출력 함수

```
    fp = fopen("data.dat", "rb");
    if (fp == NULL) {
        printf("파일 열기에 실패함\n");
        return 1;
    }

    //읽기
    fread(buf2, sizeof(int), 4, fp);

    //모니터 출력
    printf("%x %x %x %x\n", buf2[0], buf2[1], buf2[2], buf2[3]); //16진수
    printf("%d %d %d %d\n", buf2[0], buf2[1], buf2[2], buf2[3]); //10진수

    fclose(fp);  //파일 종료

    return 0;
}
```

```
ff 56 78 fa
255 86 120 250
```

바이너리 파일 입출력

- 이미지 복사하기

바이너리 파일 입출력

- 이미지 복사하기

```c
#define BUFFER_SIZE 4096 // 4KB 버퍼
int main()
{
    FILE* fin = fopen("boat.jpg", "rb");  // 읽기 모드
    FILE* fout = fopen("boat2.jpg", "wb"); // 쓰기 모드

    if (fin == NULL || fout == NULL) {
        perror("파일 열기 실패");
        return 1;
    }

    // 버퍼를 사용한 효율적인 복사
    int buf[BUFFER_SIZE];
    int bytesRead; //size_t bytesRead도 가능

    while ((bytesRead = fread(buf, sizeof(int), BUFFER_SIZE, fin)) > 0) {
        fwrite(buf, sizeof(int), bytesRead, fout);
    }

    fclose(fin);
    fclose(fout);
```