

## C - 파일 입출력

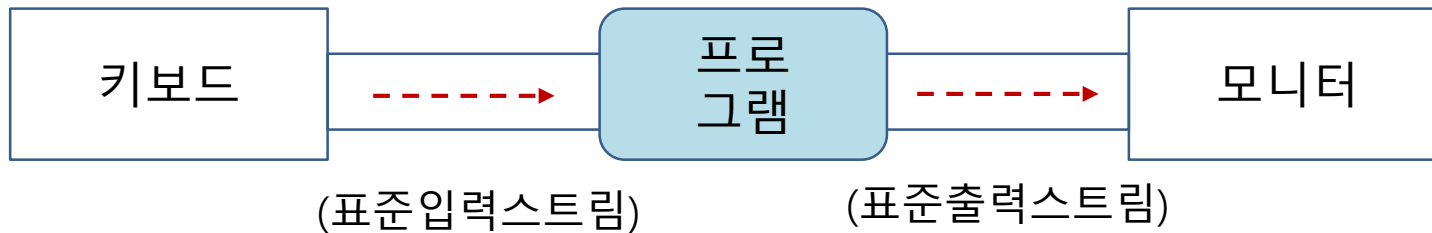
*file 10*



# 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

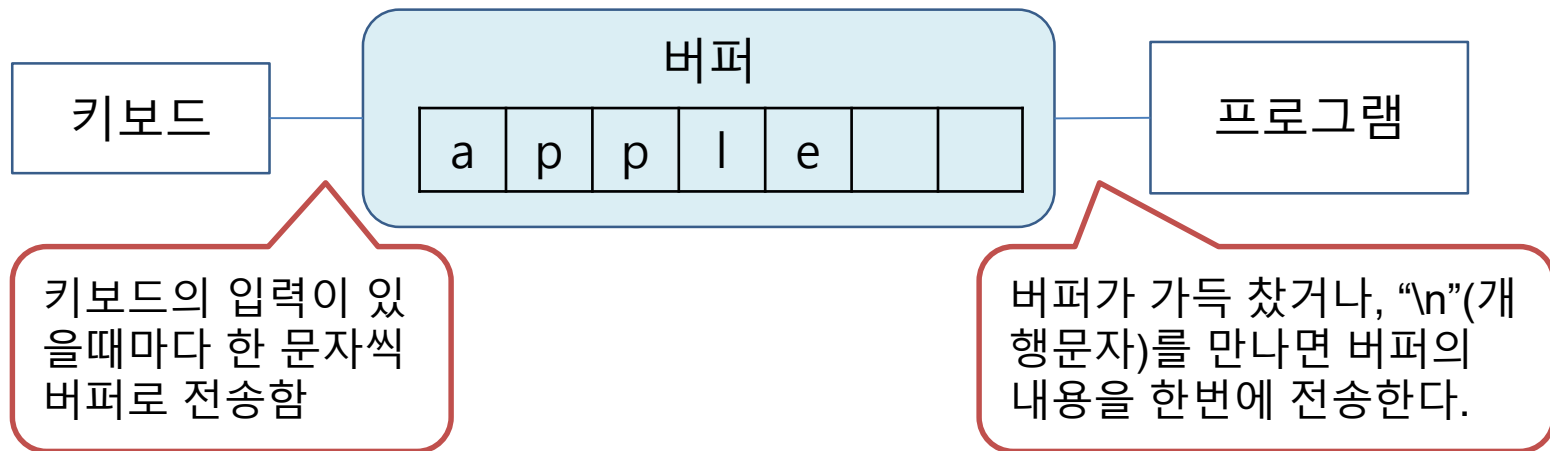


# 버퍼(Buffer)

- 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'		
---	---	---	------	--	--

fgets()는 '\n'을 가져가버려  
자동으로 출력된다.

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

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

- ▶ 해결 방법

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



# 표준 입출력

- 입력 버퍼의 동작

```
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);
```

```
학 번   입 력   : 101
이름   입 력   : 신 유 빈
학 번   : 101
이름   : 신 유 빈
```



# 파일 입출력 관련 함수

- 파일 입출력 함수 - 헤더파일 <stdio.h> 포함

함수의 원형	기능 설명
<b>fputc</b> (int c, FILE* stream)	파일에 문자 1개 쓰기
<b>fputs</b> (const char* s, FILE* stream)	파일에 문자열 쓰기
<b>fgetc</b> (FILE* stream)	파일에서 한 문자 읽기, 파일의 끝에 도달
<b>fgets</b> (char* s, int MaxCount, FILE* stream)	파일로부터 문자열 입력 받음
<b>fprintf</b> (FILE* stream, const char* format)	파일로부터 자료형에 맞춰 데이터를 입력
<b>fscanf_s</b> (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
<b>fopen</b> (const char* filename, const char* mode)	<b>fopen</b> (파일, "w")	파일에 쓴다.
	<b>fopen</b> (파일, "r")	파일을 읽는다.
	<b>fopen</b> (파일, "a")	파일에 추가로 쓴다.
<b>fclose</b> (FILE* stream)		파일을 닫음





# 파일 입출력

## ➤ 파일 쓰기

```
#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

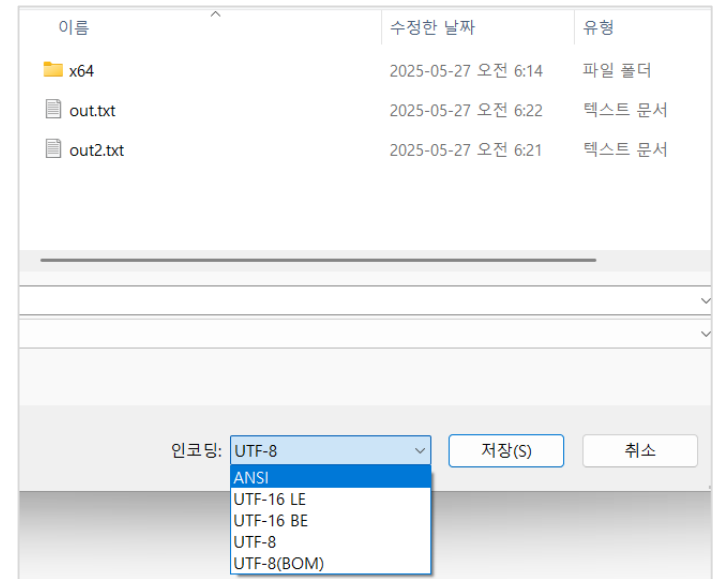
## ➤ 파일 쓰기

```
//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

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

//방법 2
/*while ((ch = fgetc(fp)) != -1) {
    printf("%c", ch);
}*/
```



# 파일 입출력

## ➤ 아스키 파일 쓰기

```
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);
}

fclose(fp);
```

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



# 파일 입출력

## ➤ 파일 쓰기(추가 저장)

```
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~



# 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

```
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

```
void wordRead()
{
    FILE* fp; //파일 포인터 변수
    char word[256]; //단어를 저장할 배열 선언

    if (fopen_s(&fp, "word.txt", "r") != 0) {
        perror("파일 열기에 실패함\n");
        return 1; //에러시 1 or -1
    }
    //파일 읽기
    while (1) {
        if (fgets(word, sizeof(word), fp) == NULL)
            break;
        printf("%s", word);
    }

    /*while (fgets(word, sizeof(word), fp) != NULL) {
        printf("%s", word);
    }*/
    fclose(fp);
}
```



# 영어 타자 게임

## ➤ 프로젝트 배포 – Release 모드

```
Release x64 로컬 Windows 디버거
(전역 범위)
#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

EnglishTyping.exe

word.txt

[영어 타자 게임], 준비되면 엔터>

문 제 1

ant

ant

통 과 !

문 제 2

monkey

monkey

통 과 !

문 제 3

dog

dog

통 과 !

문 제 4

lion

lion

통 과 !



# 영어 타자 게임

```
//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");
```

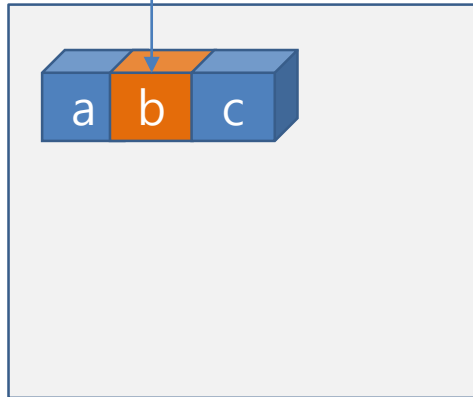


# 파일 복사

## ➤ 파일 복사 – 파일 읽고 쓰기

`fopen_s(&fin, 원본파일, "r")`

`fin`



`fgetc()`

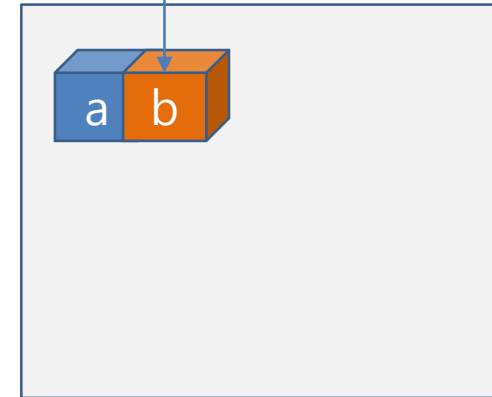


`fputc()`



`fopen_s(&fout, 복사파일, "w")`

`fout`



`fin`은 원본파일을 가리키는 파일 포인터이며, `fout`은 새로 생성할 파일을 가리키는 파일 포인터로 두 개의 파일을 오픈하게 된다.



# 파일 복사

## ➤ 파일 복사 – 읽고 쓰기

```
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);
```

ascii.txt

2025-05-29 오전 11:19

ascii2.txt

2025-05-29 오전 11:59



# 바이너리 파일 입출력

- 바이너리 파일 입출력 함수

모드 구분	기능 설명
<b>fopen(파일, "wb")</b>	바이너리 파일에 쓴다.
<b>fopen(파일, "rb")</b>	바이너리 파일을 읽는다.
<b>fopen(파일, "ab")</b>	바이너리 파일에 추가로 쓴다.

함수의 원형	기능 설명
<b>fread(버퍼, )</b>	바이너리 파일 읽기
<b>fwrite(const void* buffer, FILE* stream)</b>	바이너리 파일 쓰기





# 바이너리 파일 입출력

- 바이너리 파일 입출력

```
#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

```
#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);
}
```



# 파일 입력 받기

## ➤ 입력받아 저장하기 - 1명 입력

```
FILE* fp;
char name[20]; //이름
int eng, math; //영어, 수학 점수

//파일 입력
/*printf("이름 입력: ");
scanf_s("%s", name, sizeof(name));
printf("영어 점수 입력: ");
scanf_s("%d", &eng);
printf("수학 점수 입력: ");
scanf_s("%d", &math);*/

//콘솔 입력(키보드)
printf("이름 입력: ");
fscanf_s(stdin, "%s", name, sizeof(name));

printf("영어 점수 입력: ");
fscanf_s(stdin, "%d", &eng);
```

```
이름 입력 : 한강
영어 점수 입력 : 95
수학 점수 입력 : 87
한강 95 87
```

score.txt

한강 95 87



# 파일 입력 받기

## ➤ 입력받아 저장하기 - 1명 입력

```
printf("수학 점수 입력: ");
fscanf_s(stdin, "%d", &math);

//파일에 쓰기
fopen_s(&fp, "score.txt", "w");

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

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

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

fclose(fp);
```



# 파일 입출력

## ➤ score 파일 읽어오기

```
FILE* fp;
char name[20]; //이름
int eng, math; //영어, 수학 점수

fopen_s(&fp, "score.txt", "r");

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

//파일에서 읽어오기
//fscanf_s(fp, "%s", name, sizeof(name)); //이름만 읽기
fscanf_s(fp, "%s %d %d", name, sizeof(name), &eng, &math);

//모니터에 쓰기
//fprintf(stdout, "%s\n", name); //이름만 쓰기
fprintf(stdout, "%s %d %d\n", name, eng, math);

fclose(fp);
```



# 성적 관리

## ➤ 성적 리스트 만들기

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

scorelist.txt

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





# 성적 관리

## ➤ 성적 리스트 파일 쓰기

```
FILE* fp;
char name[20]; //이름
int no, eng, math; //학번, 영어, 수학점수

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

//제목행 쓰기
fprintf(fp, "번호 이름 영어 수학\n");
```



# 성적 관리

## ➤ 성적 리스트 파일 쓰기

```
while (1)
{
    printf("번호 입력(0이하 종료): ");
    scanf_s("%d", &no);
    if (no <= 0) break;

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

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

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

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



# 성적 관리

## ➤ 성적 리스트 파일 읽기

```
FILE* fin, * fout;
char line[100]; // 파일에서 한 줄을 저장할 버퍼

if (fopen_s(&fin, "scorelist.txt", "r") != 0) {
    perror("scorelist1.txt 파일 열기에 실패함");
    return 1;
}

if (fopen_s(&fout, "scorelist2.txt", "w") != 0) {
    perror("scorelist2.txt 파일 열기에 실패함");
    fclose(fin); // 입력 파일 닫기
    return 1;
}

// 파일 내용 한 줄씩 읽어서 다른 파일에 쓰기
while (fgets(line, sizeof(line), fin) != NULL) {
    // 줄바꿈 문자는 line에 포함되어 있으므로 따로 추가하지 않음
    fprintf(fout, "%s", line);
}
fclose(fin);
fclose(fout);
printf("scorelist1.txt의 내용이 scorelist2.txt에 저장되었습니다.\n");
```

scorelist2.txt

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



# 성적 관리

## ➤ 성적 리스트 읽어 계산하기

```
FILE* fin, * fout;
char line[100]; // 파일에서 한 줄을 저장할 버퍼
char name[20];
int no, eng, math;
float avg;

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

if (fopen_s(&fout, "scorelist3.txt", "w") != 0) {
    perror("scorelist2.txt 파일 열기에 실패함");
    fclose(fin); // 입력 파일 닫기
    return 1;
}

// 제목행 쓰기 (출력 파일에 추가)
fprintf(fout, "번호 이름 영어 수학 평균\n");
```

scorelist3.txt

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 리스트 읽어 계산하기

```
// 제목행 쓰기 (출력 파일에 추가)
fprintf(fout, "번호 이름 영어 수학 평균\n");

// 첫 번째 줄(제목행)은 무시
fgets(line, sizeof(line), fin);

// 파일 내용 한 줄씩 읽어서 평균 계산 및 저장
while (fgets(line, sizeof(line), fin) != NULL) {
    //4개의 변수와 같으면 line을 읽음
    if (sscanf_s(line, "%d %s %d %d", &no, name, sizeof(name), &eng, &math) == 4) {
        avg = (float)(eng + math) / 2; // 평균 계산
        fprintf(fout, "%3d %7s %3d %3d %6.2f\n", no, name, eng, math, avg); //파일쓰기
    }
    else {
        printf("잘못된 형식의 데이터: %s", line);
    }
}
fclose(fin);
fclose(fout);
```



# 실습 문제 – 파일에 구구단 쓰기

아래의 코드 작성 부분을 완성하여 구구단을 파일에 저장하세요.

```
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

