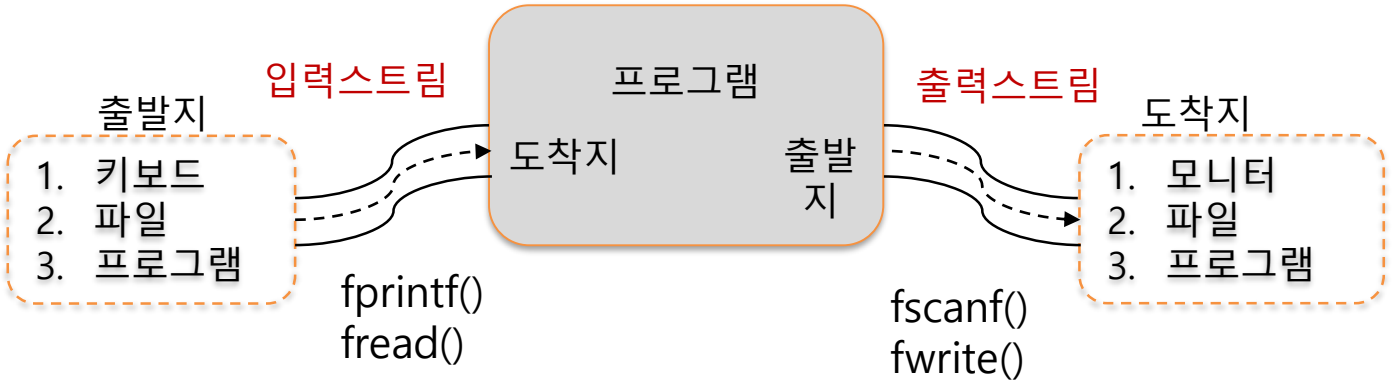
C – 파일 입출력

file IO

입, 출력 스트림

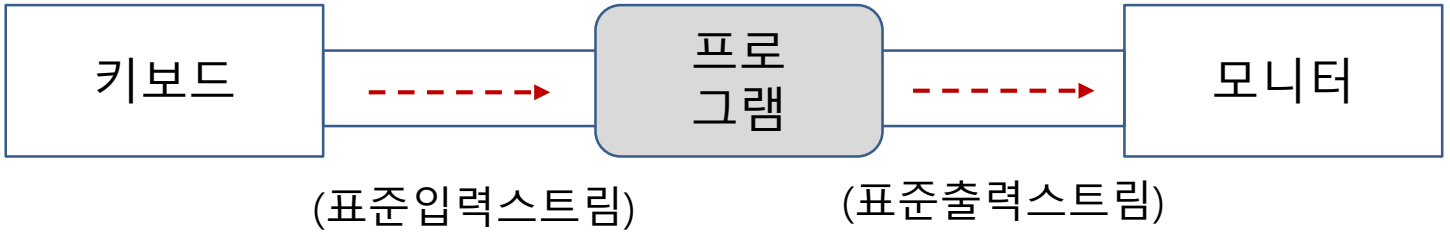
- **입, 출력 스트림**
  - **스트림**이란? 자료흐름이 물의 흐름과 같다는 뜻이다.
  - 입출력 장치와 무관하고 일관성 있게 프로그램을 구현할 수 있도록 일종의 가상통로인 스트림을 제공
  - 자료를 읽어 들이려는 소스(source)와 자료를 쓰려는 대상(target)에 따라 함수 제공됨
    입력 스트림 – 어떤 동영상을 재생하기 위해 동영상 파일에서 자료를 읽을 때 사용함
    출력 스트림 – 편집 화면에 사용자가 쓴 글을 파일에 저장할 때는 출력 스트림 사용함



스트림(stream)

- **표준 입, 출력 스트림(Stream)**

  키보드와 모니터로 입력과 출력을 수행할 때 사용되는 스트림이다.



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

파일 입출력

- **파일 입출력의 필요성**
  - 프로그램 실행 중에 메모리에 저장된 데이터는 프로그램이 종료되면 사라진다.
  - 데이터를 프로그램이 종료된 후에도 계속해서 사용하려면 파일에 저장하고 필요할 때 파일을 읽어서 데이터를 사용할 수 있다.

- **C언어에서 파일 입출력**

  프로그램이 파일(File)과 데이터를 주고 받는 기능이다.

  printf() / scanf()로 화면이나 키보드와 주고받던 것을 파일로 확장한 것이다.

파일 입출력

- **파일 입출력의 프로세스(process)**

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

- **FILE 구조체**

  FILE* 타입 변수를 사용해 파일과 연결한다.

  FILE* fp

파일 입출력

- **텍스트 파일과 바이너리 파일**
  - 텍스트 파일 – 문자 기반 스트림으로 문자를 읽고 쓰는 파일
  - 바이너리 파일 – 바이트 기반 스트림으로 이미지, 영상을 읽고 쓰는 파일

구분	텍스트 파일	바이너리 파일
저장 방식	사람이 읽을 수 있는 문자 형태	메모리의 원본 이진 데이터
크기	더 큼 (숫자를 문자로 저장)	더 작음 (메모리 그대로)
속도	상대적으로 느림	빠름
호환성	메모장 등에서 열어보기 가능	열면 깨진 글자처럼 보임
사용 예	설정 파일, 로그	이미지, 오디오, 구조체 데이터

텍스트 파일 입출력

- **텍스트 파일 입출력 읽기/쓰기 모드**

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

텍스트 파일 입출력 관련 함수

- **텍스트 파일 입출력 함수**

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

- 텍스트 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;
    }
```

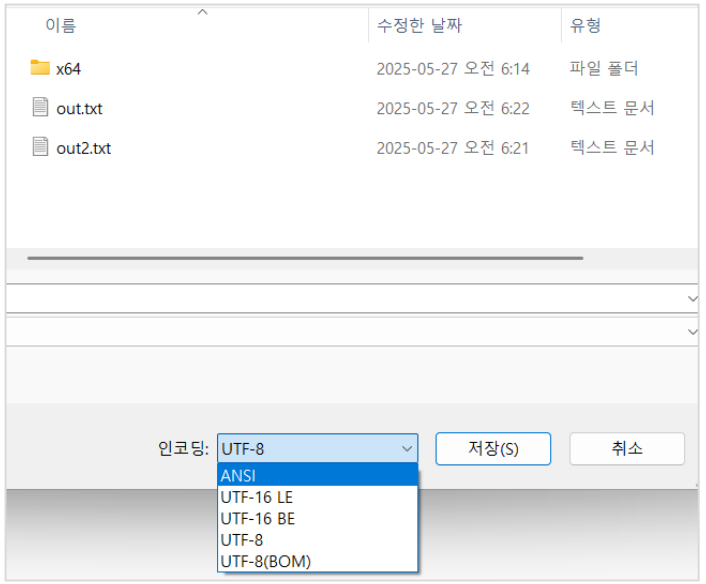
out.txt

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

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;

fp = fopen("c:/cfile/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;   //읽은 문자 변수(코드값이므로 int형)

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

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

fclose(fp);
```

파일 입출력

- 파일 쓰기(추가 저장) – append 모드

```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

- 배열로 읽기 – fgets()

```
char str[128];  //배열에 저장

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

//한 줄 읽기
fgets(str, sizeof(str), fp);
printf("%s", str);

//모든 라인 읽기
while ((fgets(str, sizeof(str), fp)) != NULL) {
    printf("%s", str);
}

fclose(fp);
```

```
Hello
Apple

좋아요
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
```

실습 문제 – 파일에 구구단 쓰기

아래의 코드 작성 부분을 완성하여 구구단을 파일에 저장하세요.

```c
FILE* fp;

fp = fopen("gugudan.txt", "w");
if (fp == NULL) {
    printf("파일 열기에 실패함\n");
    return 1;  //에러시 1 or -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
```

구조체 활용하기

- 구조체 변수 쓰고 읽기

```c
//Person 구조체
typedef struct {
    char name[20];  //이름
    int age;        //나이
} Person;

int main() {

    FILE* fp; //파일 포인터
    Person p1 = { "이정후", 26 }; //초기화
    Person p2;

    // 1. 쓰기 모드로 열기
    fp = fopen("person.txt", "w");
    if (fp == NULL) {
        printf("파일 열기 실패!\n");
        return 1;
    }
```

구조체 활용하기

- 구조체 변수 쓰고 읽기

```
    fprintf(fp, "%s %d\n", p1.name, p1.age); // 텍스트 형식으로 저장
    fclose(fp);

    // 2. 읽기 모드로 열기
    fp = fopen("person.txt", "r");
    if (fp == NULL) {
        printf("파일 열기 실패!\n");
        return 1;
    }
    fscanf(fp, "%s %d", p2.name, &p2.age); // 텍스트 형식으로 읽기
    fclose(fp);

    printf("이름: %s, 나이: %d\n", p2.name, p2.age);
    return 0;
}
```

```
이름: 이정후, 나이: 26
```

구조체 활용하기

- 구조체 배열 쓰고 읽기

```c
typedef struct{
    char name[20];
    int age;
    float height;
}Person;

int main()
{
    int size;
    printf("사람 수를 입력하세요: ");
    scanf("%d", &size);

    Person* p = (Person*)malloc(sizeof(Person) * size);

    //사용자 입력
    for (int i = 0; i < size; i++) {
        printf("이름 입력: ");
        scanf("%s", (p + i)->name);

        printf("나이 입력: ");
        scanf("%d", &(p + i)->age);
```

구조체 활용하기

- 구조체 배열 쓰고 읽기

```
    printf("키 입력: ");
    scanf("%f", &(p + i)->height);
}

// 파일에 저장
FILE* fp = fopen("data.txt", "w");

for (int i = 0; i < size; i++) {
    fprintf(fp, "%s %d %.1f\n",
        (p + i)->name, (p + i)->age, (p + i)->height);
}
fclose(fp);

// 파일에서 불러오기
fp = fopen("data.txt", "r");

for (int i = 0; i < size; i++) {
    fscanf(fp, "이름: %s, 나이: %d, 키: %f",
                (p + i)->name, &(p + i)->age, &(p + i)->height);
}
fclose(fp);
```

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

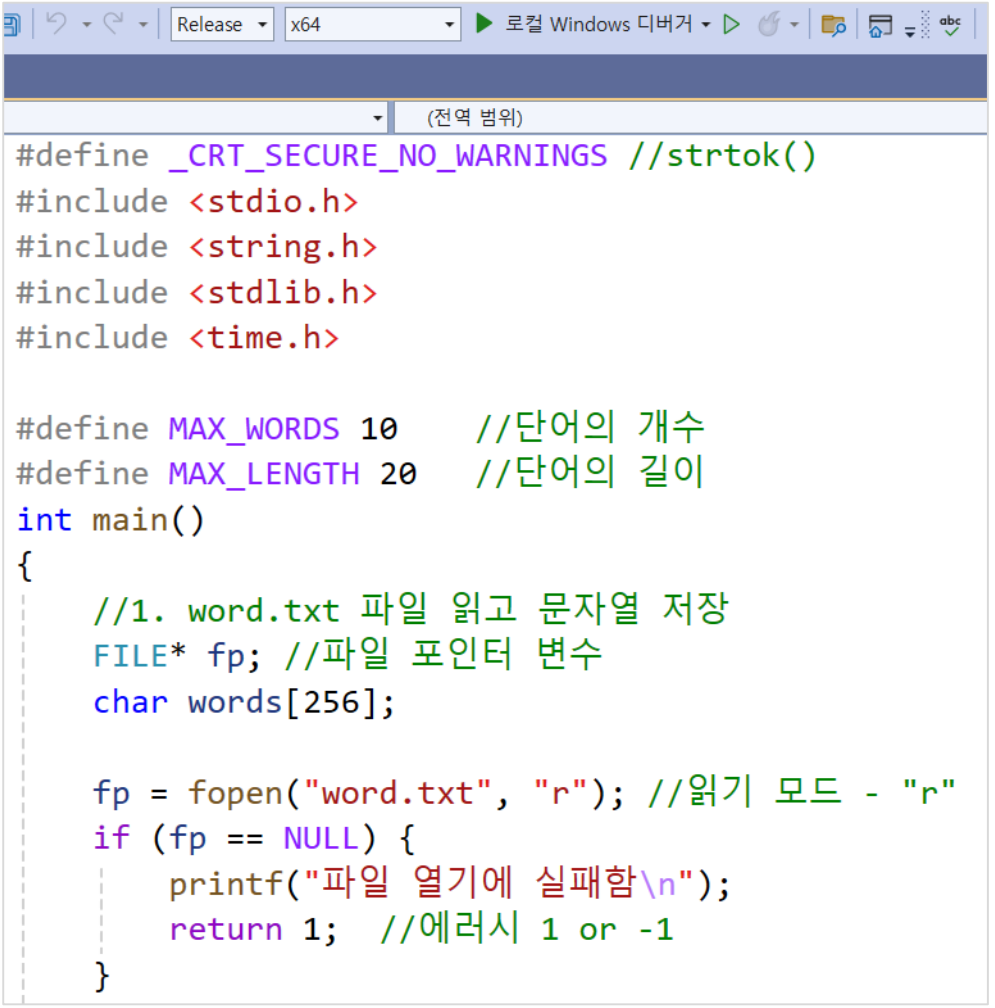
- 영어 단어 읽기 – 포인터 배열

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

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
```

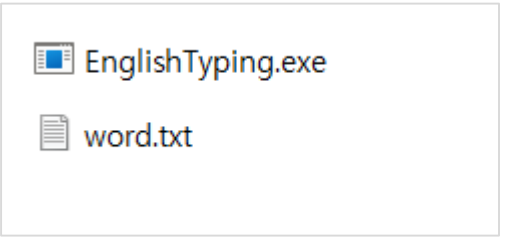
영어 타자 게임

- 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

- 타자 게임

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

// 1. 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, " ");
}
```

영어 타자 게임

- 타자 게임

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

srand(time(NULL));

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
```

영어 타자 게임

- 타자 게임

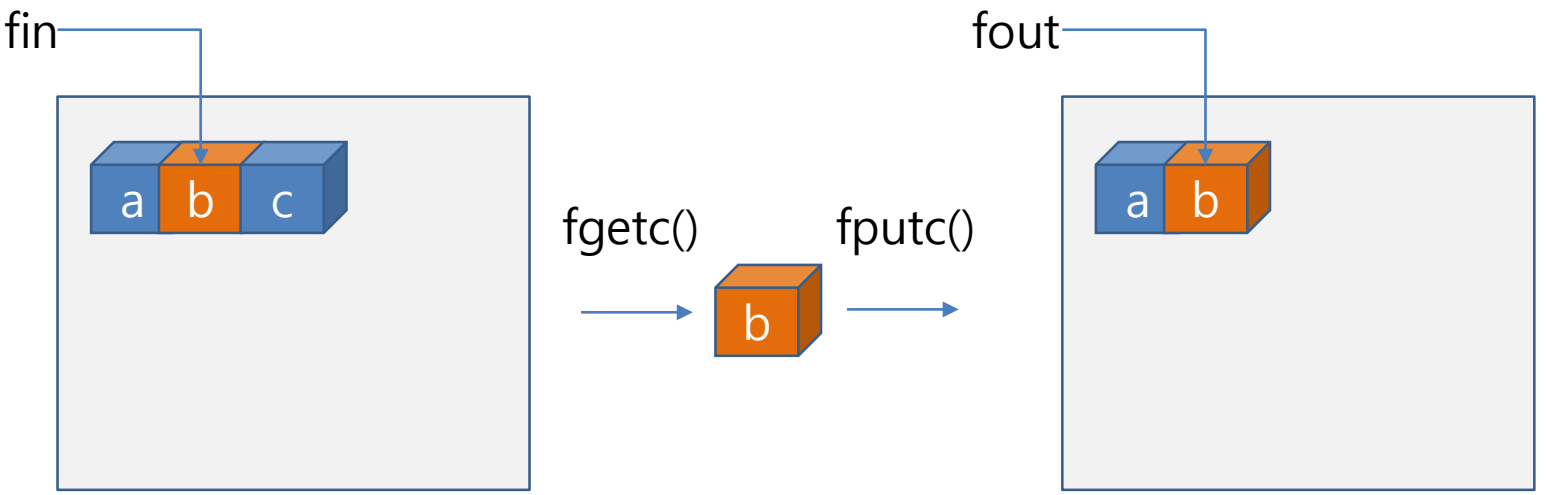
```
while (n <= 10) {
    printf("\n문제 %d\n", n);
    int rndIdx = rand() % idxOfWords; //실제 단어 개수 사용
    question = wordList[rndIdx];  //랜덤한 단어 추출
    printf("%s\n", question); //문제 출제

    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
```

파일 복사

- 파일 복사 – 파일 읽고 쓰기

fopen_s(&fin, 원본파일, "r")

fopen_s(&fout, 복사파일, "w")

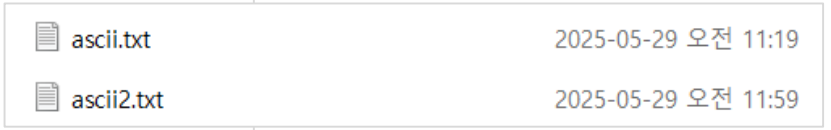


fin은 원본파일을 가리키는 파일 포인터이며, fout은 새로 생성할 파일을 가리키는 파일 포인터로 두 개의 파일을 오픈하게 된다.

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);
```



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

```
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(배열, 배열의 크기, 요소의 개수, 파일 포인터)
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
for (int i = 0; i < 4; i++) {
    printf("%x %d\n", buf2[i], buf2[i]); //16진수 주소, 값
}

fclose(fp);  //파일 종료
```

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

바이너리 파일 입출력

- 바이너리 파일 입출력 – 구조체 사용

```c
typedef struct {
    char name[20];
    int age;
} Person;

int main() {
    FILE* fp;
    Person p1 = { "이정후", 26 };
    Person p2;

    // 1. 쓰기 모드로 열기 (바이너리)
    fp = fopen("person.dat", "wb");
    if (fp == NULL) {
        printf("파일 열기 실패!\n");
        return 1;
    }
```

바이너리 파일 입출력

- 바이너리 파일 입출력 – 구조체 사용

```
    fwrite(&p1, sizeof(Person), 1, fp); // 구조체 그대로 저장
    fclose(fp);

    // 2. 읽기 모드로 열기 (바이너리)
    fp = fopen("person.dat", "rb");
    if (fp == NULL) {
        printf("파일 열기 실패!\n");
        return 1;
    }
    fread(&p2, sizeof(Person), 1, fp); // 구조체 그대로 읽기
    fclose(fp);

    printf("읽은 데이터 → 이름: %s, 나이: %d\n", p2.name, p2.age);
    return 0;
}
```

바이너리 파일 입출력

- 이미지 복사하기

boat1.jpg

boat2.jpg

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);
```