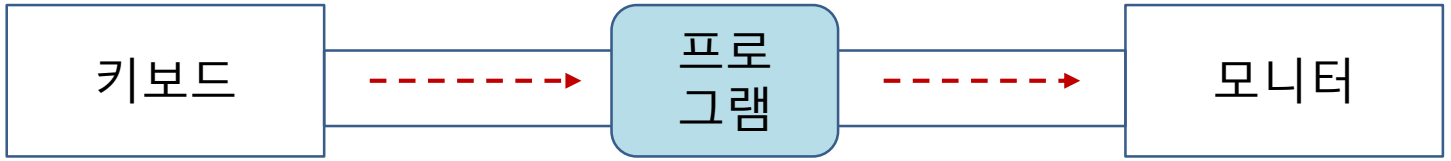
파일 입출력 / 파일 배포

Visual Studio 2022

파일 입출력

➢ **스트림이란(Stream)?**

데이터를 입력하고 출력하기 위한 연결 통로이다.



표준입력스트림 표준출력스트림

✓ 기본개방 스트림파일 -> 메모리에 구성한 논리적 파일(물리적 아닌)
운영체제가 제어 및 관리함

스트림	설명	장치
stdin	표준 입력을 담당	키보드
stdout	표준 출력을 담당	모니터
stderr	표준 에러를 담당	모니터

파일 입출력

- **버퍼와 버퍼링**

버퍼(Buffer)는 처리할 데이터를 임시로 저장하는 저장소

입력버퍼는 데이터를 저장하기 위한 버퍼이며, 출력 버퍼는 데이터를 출력하기 위한 버퍼이다.

- **콘솔 입출력 함수 및 파일 입출력 함수**

콘솔은 키보드나 모니터와 같은 표준 입출력 장치이다.

콘솔 입출력 함수

getchar(), putchar(), gets(), puts(), printf(), scanf() 등

파일 입출력 함수

fgetc(), fputc(), fgets(), fputs(), fscanf(), fprintf() 등

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.

파일을 이용한 입출력 과정

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

파일 입출력

- 파일 입출력 함수 - 헤더파일 <stdio.h> 포함

함수의 원형	기능 설명
fopen(const char* filename, const char* mode**)**	파일스트림을 생성하고 파일을 연다.
fclose(FILE* stream**)**	파일을 닫음
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
sscanf_s(const char* str, const char* format, ..)	str문자열에서 format형식으로 데이터를 읽음

파일 입출력

- 파일 쓰기

fopen(파일이름, “w”) – “w”는 쓰기 모드

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

int main() {
    //스트림 생성
    FILE* fp;

    //모드 - 'r', 'w', 'a'
    fp = fopen("out.txt", "w");

    if (fp == NULL) {
        printf("파일 열기에 실패함\n");
        return 0;
    }
```

out.txt

```
Hello
Apple

사과
```

파일 입출력

- 파일 쓰기

fopen(파일이름, “w”) – “w”는 쓰기 모드

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

    //문자열 쓰기
    fputs("\nApple\n", fp);
    fputs("\n사과\n", fp);  //한글 저장

    fclose(fp);
    return 0;
}
```

파일 입출력

- 파일 읽기

fopen(파일이름, “r”) – “r”는 읽기 모드

```c
FILE* fp;  //파일 포인터 객체 생성
int ch;

fp = fopen("out.txt", "r");

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

//fgetc(FILE*) 파일에서 한글자 읽어오는 함수
/*ch = fgetc(fp);
printf("%c", ch);*/
```

파일 입출력

- 파일 읽기

fopen(파일이름, “r”) – “r”는 읽기 모드

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

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

fclose(fp);
```

파일 입출력

- 아스키 파일 쓰기

```
FILE* fp;

//fp = fopen("ascii.txt", "w");
fopen_s(&fp, "ascii.txt", "w"); //_CRT_SECURE_NO_WARNINGS 불필요.

if (fp == NULL) {
    printf("파일 열기에 실패함\n");
    return 0;
}

printf("=== ASCII 테이블을 작성합니다. ===");
//아스키 코드 - 31번까지 제어문자, 32-공백문자
for (int i = 32; i < 128;i++) {
    if (i % 10 == 0)
        fputc('\n', fp);
    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

- 파일 쓰기(추가 저장)

fopen(파일이름, “a”) – “a”는 추가 쓰기 모드

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

//'a' -> 추가 모드
fp = fopen("out.txt", "a");

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

//문자열 쓰기
fputs("Good Luck~\n", fp);

fprintf(fp, "%s\n", msg);

fclose(fp);
```

out.txt

```
Hello
Apple

사과
Good Luck~
행운을 빌어요~
```

실습 문제

- 구구단 파일 쓰기

```c
int i, j;
FILE* fp;
fp = fopen("gugudan.txt", "w");

for (i = 2; i < 10; i++) {
    for (j = 1; j < 10; j++) {
        fprintf(fp, "%d x %d = %d\n", i, j, (i * j));
    }
    fprintf(fp, "\n");
}

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
```

실습 문제

- 파일 쓰기 및 읽기

```c
#include <stdio.h>

void writeFile();
void readFile();
int main() {

    writeFile();  // 파일 쓰기 함수 호출

    readFile();   // 파일 읽기 함수 호출

    return 0;
}
```

data.txt

```
Hello, File I/O!
100
```

실습 문제

- 파일 쓰기 및 읽기

```c
// 파일 쓰기 함수
void writeFile() {
    FILE* fp = fopen("data.txt", "w"); // 쓰기 모드로 파일 열기
    int n = 100;

    if (fp == NULL) {
        printf("파일을 열 수 없습니다.\n");
        exit(1);
    }

    fprintf(fp, "Hello, File I/O!\n"); // 파일에 문자열 쓰기
    fprintf(fp, "%d\n", n);       // 숫자 쓰기

    fclose(fp); // 파일 닫기
    printf("파일 쓰기 완료!\n");
}
```

실습 문제

- 파일 쓰기 및 읽기

```c
// 파일 쓰기 함수
void writeFile() {

    FILE* fp = fopen("data.txt", "w"); // 쓰기 모드로 파일 열기
    int n = 100;

    if (fp == NULL) {
        printf("파일을 열 수 없습니다.\n");
        exit(1);
    }

    fprintf(fp, "Hello, File I/O!\n"); // 파일에 문자열 쓰기
    fprintf(fp, "%d\n", n);       // 숫자 쓰기

    fclose(fp); // 파일 닫기
    printf("파일 쓰기 완료!\n");
}
```

실습 문제

- 파일 쓰기 및 읽기

```c
void readFile() {
    FILE* fp = fopen("data.txt", "r");
    int ch; //읽은 데이터(코드값) 변수
    char buffer[100]; // 읽은 데이터 배열

    if (fp == NULL) {
        printf("파일을 열 수 없습니다.\n");
        exit(1);
    }

    printf("===== 파일 내용 출력 =====\n");
    //문자 1개씩 읽기
    /* while ((ch = fgetc(fp)) != EOF)
    {
        printf("%c", ch);
    }*/
    //배열로 읽기
    while (fgets(buffer, sizeof(buffer), fp) != NULL) {
        printf("%s", buffer);
    }
    fclose(fp); // 파일 닫기
}
```

파일 입출력

- 영어 단어 쓰기

```c
FILE* fp; // 쓰기 모드 ("w")

if (fopen_s(&fp,"words.txt", "w") != 0) {
    perror("파일 열기에 실패했습니다!\n");
    return 1;
}

// 영어 단어 목록
char* words[] = { "ant", "bear", "chicken", "cow", "dog", "elephant",
        "monkey", "lion", "tiger", "horse", "snake" };
int wordCount = sizeof(words) / sizeof(words[0]);
//printf("wordCount = %d\n", wordCount);

for (int i = 0; i < wordCount; i++) {
    fprintf(fp, "%s\n", words[i]); // 단어를 한 줄씩 저장
}

fclose(fp);
printf("영어 단어를 파일에 저장했습니다.\n");
```

word.txt

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

실습 문제

- 영어 타자 게임

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#define MAX_WORDS 11
#define MAX_LENGTH 20

int main()
{
    FILE* fp;

    if (fopen_s(&fp,"words.txt", "w") != 0) {
        perror("파일 열기에 실패했습니다!\n");
        return 1;
    }

    char words[MAX_WORDS][MAX_LENGTH]; // 단어를 저장할 배열
    int count = 0; //단어의 개수
```

실습 문제

- 영어 타자 게임

```
char words[MAX_WORDS][MAX_LENGTH]; // 단어를 저장할 배열
int count = 0; //단어의 개수

while (fgets(words[count], MAX_LENGTH, fp) != NULL) {
    // 개행 문자 제거 (fgets는 개행 문자도 포함)
    words[count][strcspn(words[count], "\n")] = '\0';
    count++;
    /*
    fgets(words[count], MAX_LENGTH, fp) : 파일에서 한줄씩
                읽어서 words[count]에 저장
    strcspn(words[count], "\n") : 개행 문자의 위치를 찾음.
    words[count][strcspn(words[count], "\n")] = '\0' : 개행 문자를
            '\0' 널문자로 바꿈
    */
}
//printf("단어의 개수: %d\n", count);
fclose(fp);
```

실습 문제

- 영어 타자 게임

```
char* question;   //문제
char answer[20];  //사용자
int n = 1;  //문제 번호
clock_t start, end;
double elapsedTime; //게임 소요 시간

srand(time(NULL)); //seed 설정
int size = sizeof(words) / sizeof(words[0]);

printf("영어 타자 게임, 준비되면 엔터>");
getchar();

start = clock();  //시작 시간

while (n <= 10)
{
    printf("\n문제 %d\n", n);
    int rndIdx = rand() % size;
    question = words[rndIdx];
```

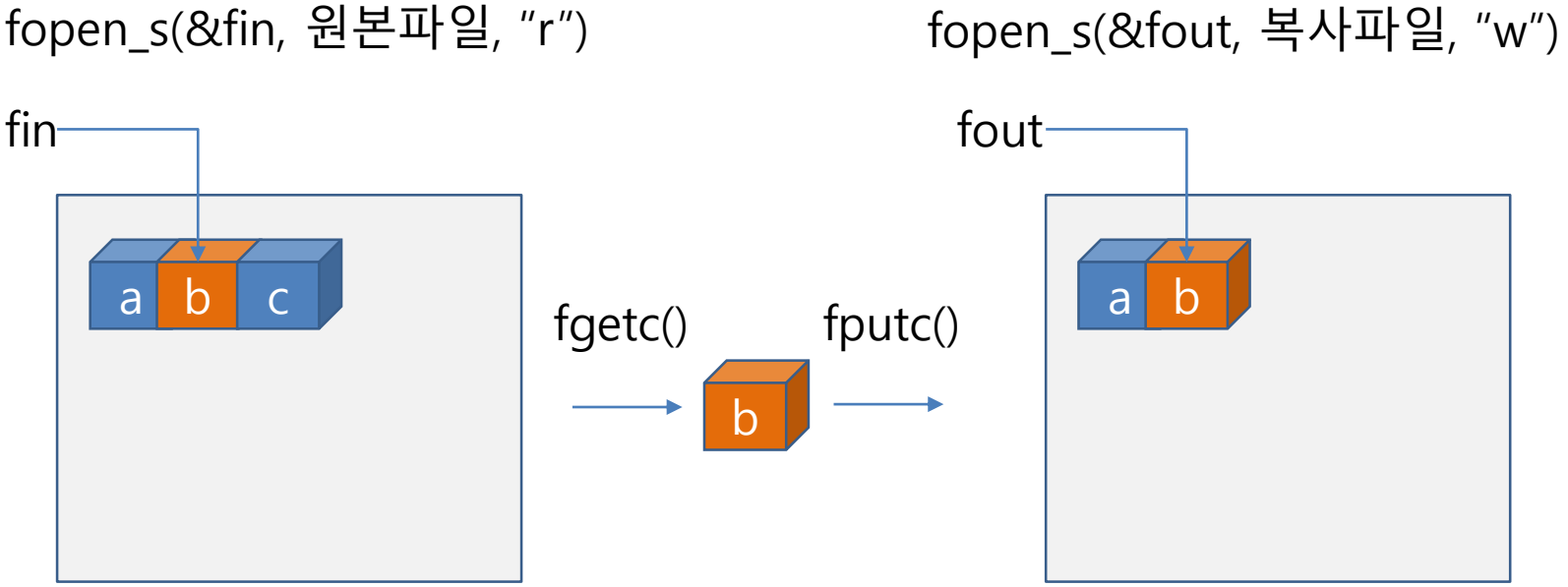
실습 문제

- 영어 타자 게임

```
    printf("%s\n", question);  //문제 출제
    scanf_s("%s", answer, sizeof(answer));  //사용자 입력

    if (strcmp(question, answer) == 0)
    {
        printf("통과!\n");
        n++;   //다음 문제
    }
    else
    {
        printf("오타! 다시 도전!\n");
    }
}
end = clock(); //종료 시간
elapsedTime = (double)(end - start) / CLOCKS_PER_SEC;
printf("게임 소요 시간: %.2lf초\n", elapsedTime);

system("pause");  //exe 파일 실행시 필수

return 0;
```

파일 복사

- 파일 복사 – 파일 읽고 쓰기



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

파일 복사

- 파일 복사 – 읽고 쓰기

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

fopen_s(&fin, "ascii.txt", "r"); //원본에서 읽기
fopen_s(&fout, "ascii_copy.txt", "w"); //복사본에 쓰기

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

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

파일 입력 받기

- 입력받아 저장하기

```
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
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
```

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

```c
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

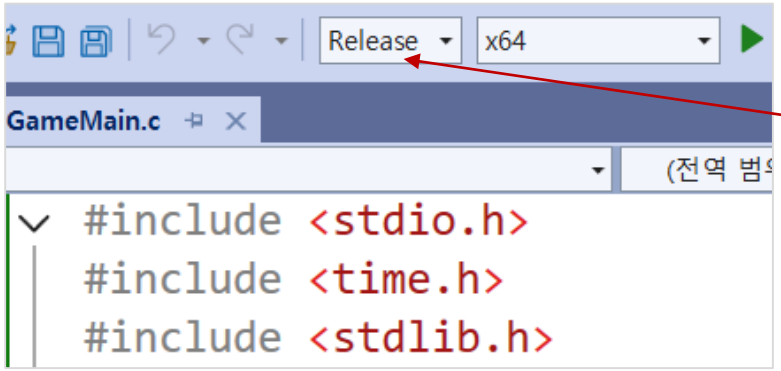
- 성적 리스트 읽어 계산하기

```
// 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);
```

파일 배포

◈ 파일 배포

파일 배포란 c언어 소스파일을 .exe 실행 파일로 만들어 공개 및 서비스 하는 것을 말한다.

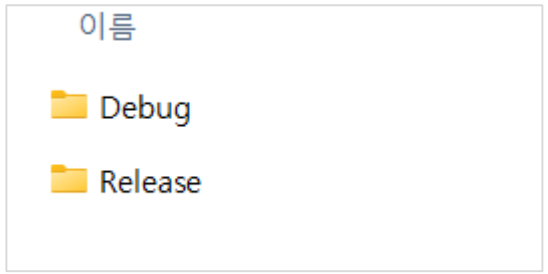


Debug 모드를
Release 모드로 바꾼다.

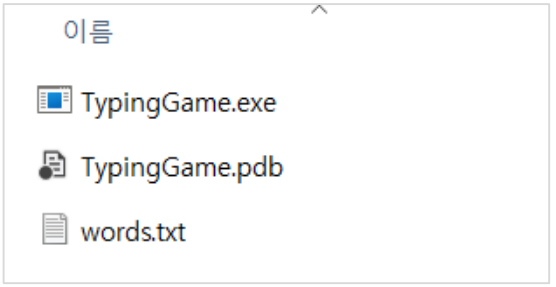
Ctrl + F5로 실행

파일 배포

◈ 파일 배포



Release 폴더 생성됨



TypingGame.exe 파일 생성
words.txt 추가해 줌

파일 배포

◈ exe 파일이 실행되지 않는 문제 해결

system(“pause”) 를 명시함

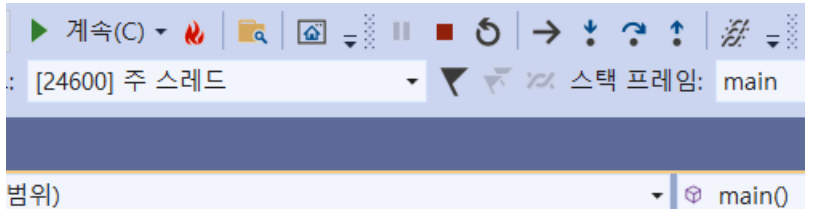
```
    else
    {
        printf("오타! 다시 도전!\n");
    }
}
end = clock(); //종료 시간
elapsedTime = (double)(end - start) / CLOCKS_PER_SEC;
printf("게임 소요 시간: %.2lf초\n", elapsedTime);

system("pause");  //exe 파일 실행시 필수
```

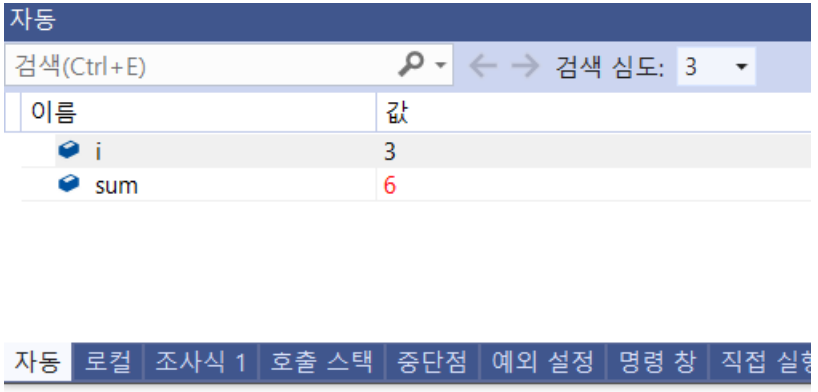
디버깅(Debugging)

◈ 디버깅 작업

- 중단점 설정(F9) : 디버그 > 중단점 설정
- 실행(F10) : 디버그 > 프로시저 단위 실행

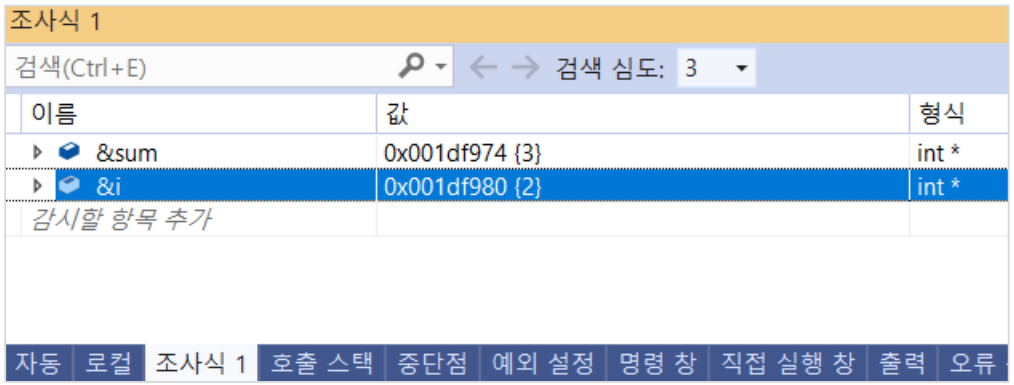
```
3   int main() {
4
5       int i = 0, sum = 0;
6
7       for (i = 1; i <= 10; i++) {
8           sum = sum + i;
9       } 경과 시간 1ms 이하
10
11      printf("합 : %d", sum);
12
13      return 0;
14  }
```





디버깅(Debugging)

◈ 디버깅 작업



주소값 확인

&sum

&i