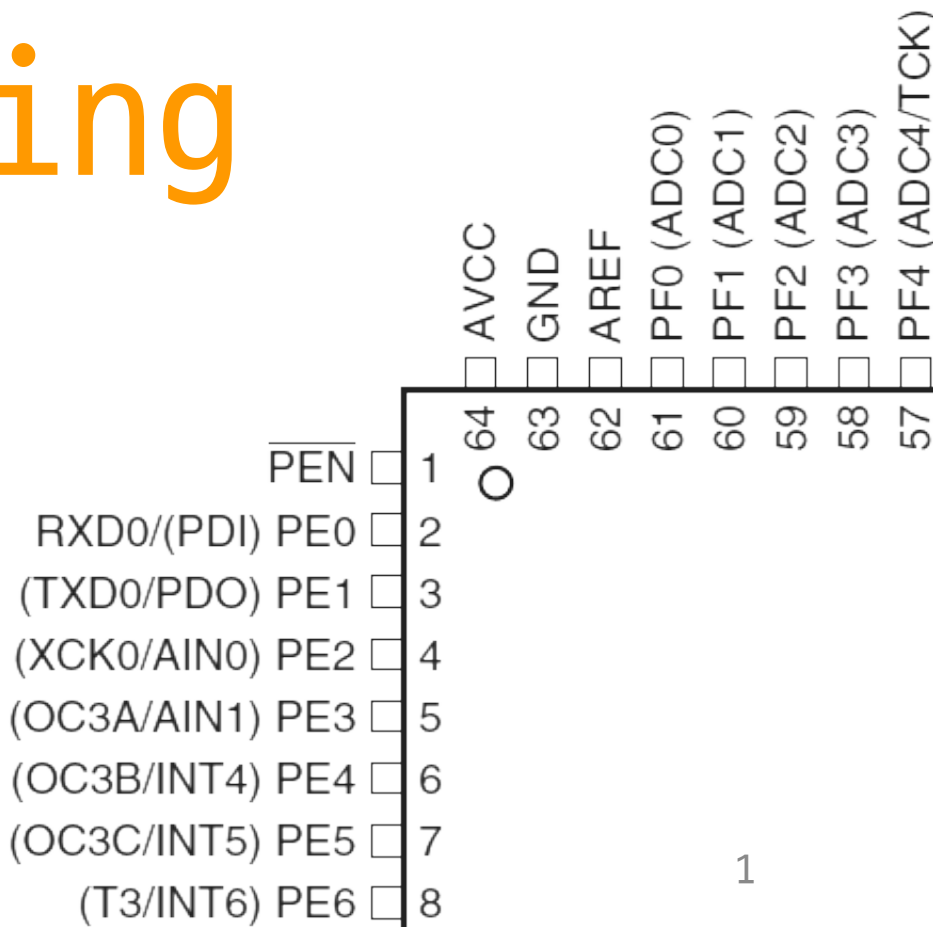


Structures and File handling



•Structure

➤여러 자료형의 변수를 묶어서 사용할 때 활용

➤이제 96명의 학생의 성적을 저장할 수 있습니다.

➤배열

➤`int std[96]`

➤그럼 최종 등급과 중간고사, 기말고사, 출석 점수를 함께 저장하려면...?

➤`char grade[96]; int mid_term[96]; int final_term[96]; int attendance[96];`

➤이렇게 프로그래밍 할 수 있을까요? 진짜?

➤"구조체"를 활용해서 다양한 자료형의 변수를 묶을 수 있음

```
struct student
{
    char grade;
    int mid_term;
    int final_term;
    int attendance;
};
```

•Structure

➤구조체의 정의 : struct 키워드를 사용

```
struct student // student라는 이름의 구조체를 정의
{
```

```
    //구조체 멤버 변수들의 정의
```

```
    char grade;
```

```
    int mid_term;
```

```
    int final_term;
```

```
    int attendance;
```

```
}; // 구조체의 정의는 반드시 ;로 끝나야 한다.
```

•Structure

➤구조체 변수의 선언

➤`struct` [구조체의 이름] [변수 이름]

➤`struct student a;`

➤ a라는 이름의 student 구조체를 선언한다.

➤구조체 멤버 변수의 접근

➤구조체 변수 이름.멤버 변수의 이름

➤`a.grade = 'A';`

➤포인터 참조의 경우

➤ 반드시 '->'로 참조 해야 함.

➤ `struct student *a; ...중략... a->grade='A';`

• 예제 1

```
#include <stdio.h>

struct student // student라는 이름의
구조체를 정의
{
    //구조체 멤버 변수들의 정의
    char grade;
    int mid_term;
    int final_term;
    int attendance;
}; // 구조체의 정의는 반드시 ;로 끝나
야 한다.
```

```
int main()
{
    struct student std1;

    std1.grade = 'A';
    std1.mid_term = 97;
    std1.final_term = 92;
    std1.attendance = 10;

    printf("std1's grade is %c\n", std1.grade);
    printf("std1's mid_term is %d\n", std1.mid_term);
    printf("std1's final_term is %d\n", std1.final_term);
    printf("std1's attendance is %d\n", std1.attendance);
}
```

•typedef

➤ 기존에 존재하는 자료형에 새로운 이름을 붙이는 것

➤ `typedef unsigned int uint;`

➤ `unsigned int` 라는 자료형에 `uint`라는 새로운 이름을 붙임

➤ 구조체도 동일하게 사용 가능

➤ `typedef struct student STD;`

➤ 앞에서 선언한 `student` 구조체에 `STD`라는 새로운 이름을 붙임

➤ 따라서 코딩할 때 `struct student` 라고 쓰는 대신, `STD`라고 쓸 수 있음.

•예제 2-1

```
#include <stdio.h>

struct student // student라는 이름의 구조체를
정의
{
    //구조체 멤버 변수들의 정의
    char grade;
    int mid_term;
    int final_term;
    int attendance;
}; // 구조체의 정의는 반드시 ;로 끝나야 한다.

typedef struct student STD;
// struct student를 STD로 쓸 수 있음
```

```
int main()
{
    STD std1; // struct student 대신 STD

    std1.grade = 'A';
    std1.mid_term = 97;
    std1.final_term = 92;
    std1.attendance = 10;

    printf("std1's grade is %c\n", std1.grade);
    printf("std1's mid_term is %d\n", std1.mid_term);
    printf("std1's final_term is %d\n", std1.final_term);
    printf("std1's attendance is %d\n", std1.attendance);
}
```

•예제 2-2

```
#include <stdio.h>

typedef struct student // student라는
이름의 구조체를 정의
{
    //구조체 멤버 변수들의 정의
    char grade;
    int mid_term;
    int final_term;
    int attendance;
}STD; // 구조체의 정의는 반드시 ;로 끝
나야 한다.
// 구조체의 정의와 typedef를 함께 진행
```

```
int main()
{
    STD std1; // struct student 대신 STD

    std1.grade = 'A';
    std1.mid_term = 97;
    std1.final_term = 92;
    std1.attendance = 10;

    printf("std1's grade is %c\n", std1.grade);
    printf("std1's mid_term is %d\n", std1.mid_term);
    printf("std1's final_term is %d\n", std1.final_term);
    printf("std1's attendance is %d\n", std1.attendance);
}
```


•구조체 배열

➤다른 자료형과 마찬가지로 구조체 역시 배열을 생성할 수 있음

➤STD student[10];

➤STD형 자료형을 10개를 가지는 student라는 구조체 배열을 생성

➤student[0].grade = 'A';

➤배열 구조체의 0번째 구조체의 멤버 변수에 접근하여 'A'를 저장

• 예제 3

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

typedef struct student
{
    char grade;
    int mid_term;
    int final_term;
    int attendance;
}STD;

int main()
{
    STD student[10];
    int fi;
    int sum;

    srand((unsigned)time(NULL));
```

```
    for (fi = 0; fi < 10; fi++)
    {
        student[fi].mid_term = rand() % 46;
        student[fi].final_term = rand() % 46;
        student[fi].attendance = rand() % 11;
        sum = student[fi].mid_term + student[fi].final_term + student[fi].attendance;

        if (sum > 80)
            student[fi].grade = 'A';
        else if (sum > 70)
            student[fi].grade = 'B';
        else if (sum > 60)
            student[fi].grade = 'C';
        else if (sum > 50)
            student[fi].grade = 'D';
        else
            student[fi].grade = 'F';

        printf("student %d's grade is %c\n", fi, student[fi].grade);
        printf("student %d's total grade is %d\n", fi, sum);
    }
}
```

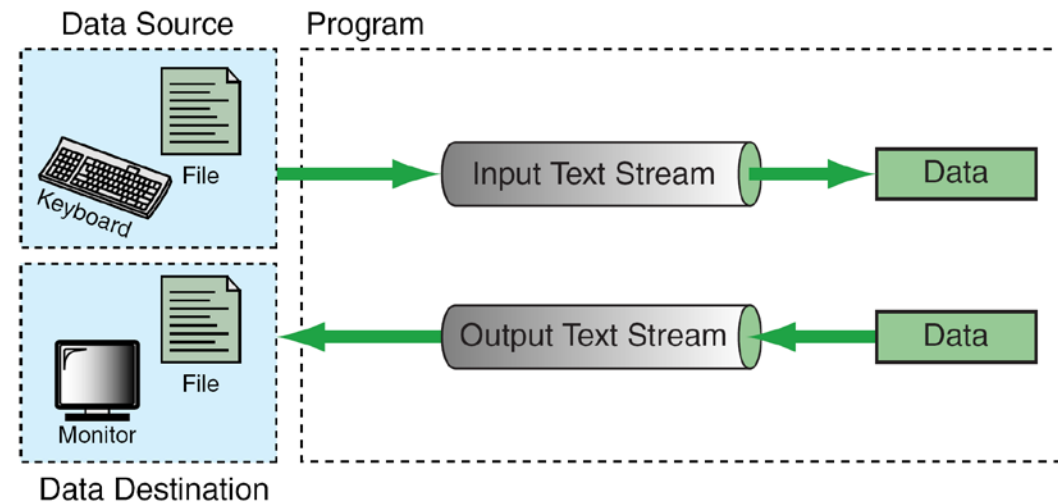
• 입출력

➤ 프로그램의 모든 데이터 입출력은 stream을 통해서 구현됨

➤ 데이터 입력을 위한 입력 스트림

➤ 데이터 출력을 위한 출력 스트림

➤ 그 동안 scanf, printf는 다 입/출력 스트림을 통해서 데이터를 주고 받았음.



•파일 입출력 - fopen

- 프로그램의 모든 데이터 입출력은 stream을 통해서 구현됨
 - 따라서 파일의 입출력 역시 stream을 생성해야 가능함
 - fopen 함수를 사용하여 stream을 생성 할 수 있음.
 - `FILE *fopen(const char *filename, const char *mode)`
 - `FILE *fp; // fopen 함수의 반환형은 FILE 구조체의 포인터다.`
 - `fp = fopen("test_text.txt", "wt"); // test_text.txt를 wt 모드로 연다.`
 - 생성한 스트림은 반드시 닫아줘야 한다.
 - 더 이상 해당 파일에 대한 입출력이 발생하지 않으면 반드시 파일을 닫는다.
 - `int fclose(FILE *stream);`
 - `fclose(fp);`

•파일 입출력 - fopen

➤파일의 개방 모드

1. 입출력 방향에 따라 구분

1. 읽기 : r, 파일이 없는 경우, 오류 발생
2. 쓰기 : w, 파일이 없는 경우, 파일을 생성함
 - 기존에 파일이 있는 경우, 그 파일을 지우고 재생성한다.
3. 읽기+쓰기 : r+, 파일이 없는 경우, 오류 발생
4. 읽기+쓰기 : w+, 파일이 없는 경우, 파일을 생성함
 - 기존에 파일이 있는 경우, 그 파일을 지우고 재생성한다.
5. 덧붙이기 : a, 파일이 없는 경우, 파일을 생성함
 - 기존에 파일이 있는 경우, 그 끝에 덧붙여서 기록한다.

•파일 입출력 - fopen

➤파일의 개방 모드

2. 바이너리 - 텍스트 데이터에 따라 구분

1. b : 바이너리 입출력 : 이미지, 소리, 동영상 등 이진 데이터가 저장되는 파일
2. t : 텍스트 입출력 : 문자 데이터를 저장하는 파일
3. 이러한 모드 구분은 1의 데이터 입출력 구분 뒤에 붙여서 쓴다.
 - rb, rt, wb, wt....

3. 예를 들어...

- `fopen("test_text.txt", "wt");` // test_text.txt를 텍스트 쓰기 모드로 연다.
- `fopen("img.bmp", "rb");` // img.bmp를 이진 읽기 모드로 연다.

•파일 입출력 함수

➤파일 스트림에 데이터를 입/출력하는 함수들(1)

➤`int fputc(int ch, FILE *stream);`

➤file stream에 1개의 문자를 출력한다.

➤`fputc('T', fp);`

➤`int fgetc(FILE *stream);`

➤file stream에서 1개의 문자를 입력 받는다.

➤`c = fgetc(fp);`

➤`int fputs(const char *str, FILE *stream);`

➤file stream에 문자열을 출력한다.

➤`fputs(str, fp);`

➤`char *fgets(char *str, int count, FILE *stream);`

➤file stream에서 문자열을 입력 받는다.

➤`fgets(buf, 200, fp);`

• 예제 4-1

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

int main()
{
    FILE *fp;

    char *str = "The key to realizing a dream is to focus not on
success but significance - and then even the small steps and
little victories along your path will take on greater meaning.";

    char buf[200];
    int fi;

    fp = fopen("test_text.txt", "wt");
    if(fp==NULL)
    {
        printf("Fail to open file\n");
        return -1;
    }

    fputc('T', fp);
    fputc('E', fp);
    fputc('S', fp);
    fputc('T', fp);
    fputc('\n', fp);

    fputs(str, fp);

    fclose(fp);

    fp = fopen("test_text.txt", "rt");

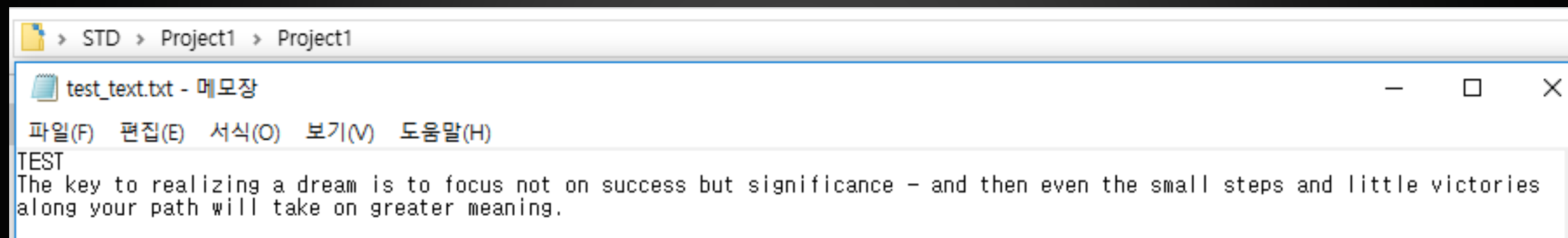
    for (fi = 0; fi < 5; fi++)
    {
        printf("%c", fgetc(fp));
    }

    fgets(buf, 200, fp);
    printf("%s", buf);
    fclose(fp);
    return 0;
}
```


•예제 4-1 실행 결과

TEST

The key to realizing a dream is to focus not on success but significance - and then even the small steps and little victories along your path will take on greater meaning.



test_text.txt 파일은 VS 프로젝트 폴더 내에 있다.

• 예제 4-2

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

int main()
{
    FILE *fp;
    char *str = "The key to realizing a dream is to focus not on
success but significance - and then even the small steps and
little victories along your path will take on greater meaning.";
    char buf[200];
    int fi;

    fp = fopen("test_text.txt", "wt");

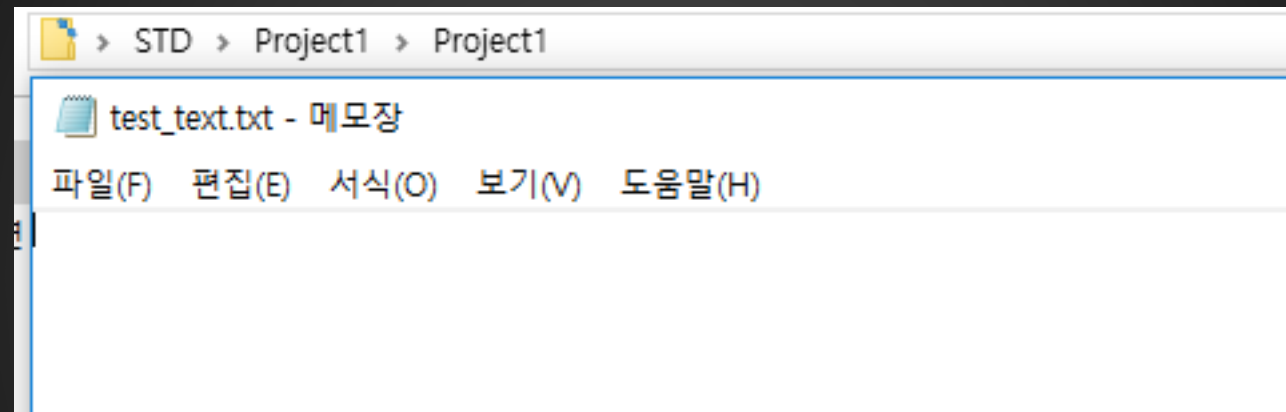
    if(fp==NULL)
    {
        printf("Fail to open file\n");
        return -1;
    }
```

```
        fputc('T', fp);
        fputc('E', fp);
        fputc('S', fp);
        fputc('T', fp);
        fputc('\n', fp); fputs(str, fp);

        fclose(fp);

        fp = fopen("test_text.txt", "wt");
        // 한번 기록한 파일을 다시 쓰기 모드로
        연다.
        fclose(fp);
        return 0;
    }
```

•예제 4-2



파일을 읽기 모드로(wt) 열면, 기존에 있던 내용이 삭제된다.

•파일 입출력 함수

➤파일 스트림에 데이터를 입/출력하는 함수들(2)

➤`int fprintf(FILE *stream, const char *format, ...);`

➤`int fscanf(FILE *stream, const char *format, ...);`

➤printf, scanf함수 처럼 형식 지정자를 통해 데이터 입출력을 할 수 있다.

• 예제 5

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

typedef struct student
{
    char grade;
    int mid_term;
    int final_term;
    int attendance;
}STD;

int main()
{
    FILE *fp;
    STD st;

    fp = fopen("student.txt", "wt");
    if (fp == NULL)
    {
        printf("Fail to open file\n");
    }

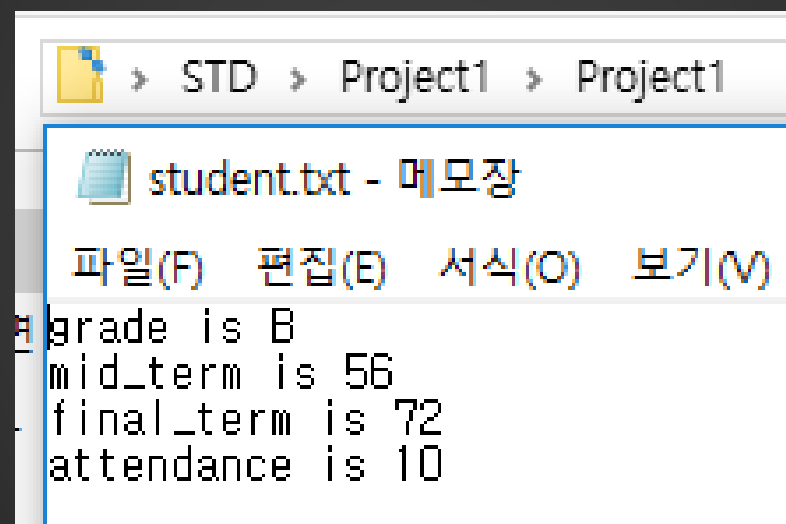
    st.grade = 'B';
    st.mid_term = 56;
    st.final_term = 72;
    st.attendance = 10;

    fprintf(fp, "grade is %c\n", st.grade);
    fprintf(fp, "mid_term is %d\n", st.mid_term);
    fprintf(fp, "final_term is %d\n", st.final_term);
    fprintf(fp, "attendance is %d\n", st.attendance);

    fclose(fp);

    return 0;
}
```

•예제 5 실행 결과



```
> STD > Project1 > Project1
student.txt - 메모장
파일(F)  편집(E)  서식(O)  보기(V)
grade is B
mid_term is 56
final_term is 72
attendance is 10
```

•파일 입출력 함수

➤파일 스트림에 데이터를 입/출력하는 함수들(3)

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

- 스트림에서 size byte씩 count번 읽어서 buffer가 지시하는 메모리에 저장한다.
- 이때, 총 몇 번 읽었는지 반환한다.
- `count = fread(buffer, 1, BUF_SIZE, ori);`

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

- printf, scanf함수 처럼 형식 지정자를 통해 데이터 입출력을 할 수 있다.
- 이때 총 몇 번 기록했는지 반환한다.
- `fwrite(buffer, 1, BUF_SIZE, cp);`

•예제 6

- 각자 원하는 파일을 선택해서 복사해보세요! ori.bmp 대신 다른 것을 입력하면 됩니다! 확장자(.bmp, .jpg 등...) 잘 확인하세요!!

```
#include <stdio.h>
#include <stdlib.h>
#define BUF_SIZE 100

int main()
{
    FILE *ori;
    FILE *cp;
    char buffer[BUF_SIZE];
    int count;

    ori = fopen("ori.bmp",
"rb");
    cp = fopen("copy.bmp",
"wb");
```

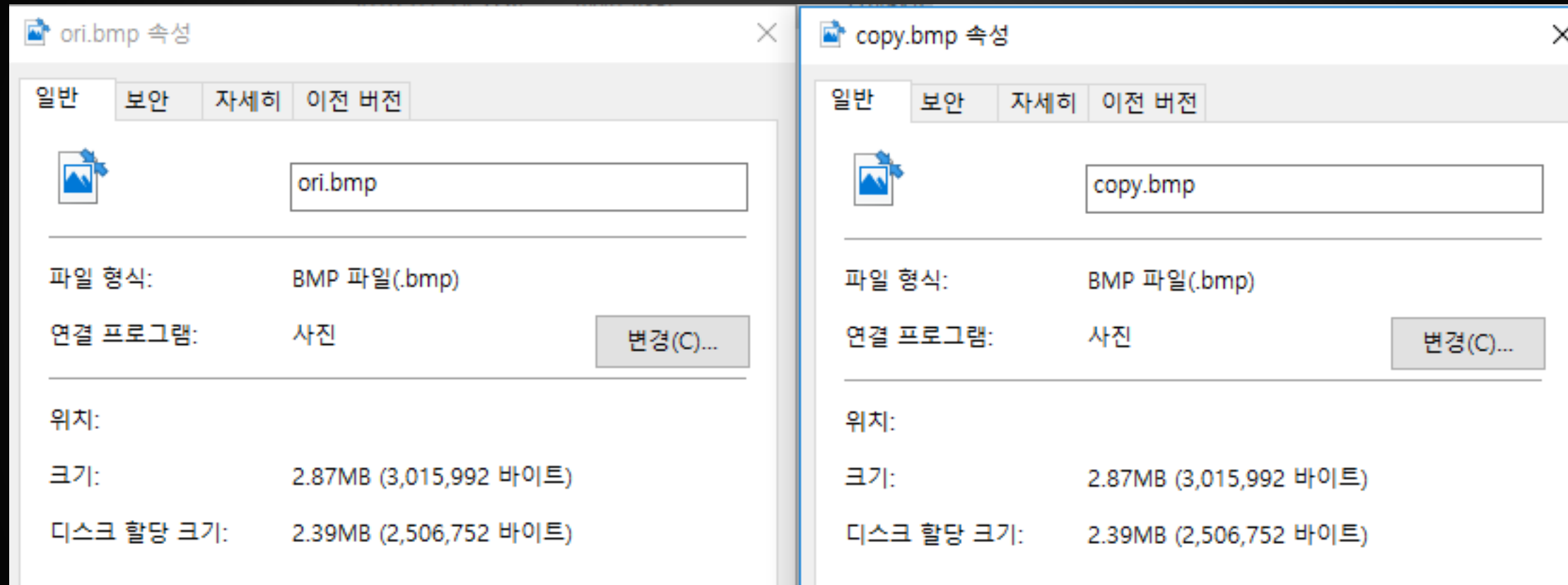
```
    if (ori == NULL || cp == NULL)
    {
        printf("Fail to open file\n");
        return -1;
    }

    while (1)
    {
        count = fread(buffer, 1, BUF_SIZE, ori); // 1바
        이트씩 BUF_SIZE만큼 파일을 읽는다.
        if (count < BUF_SIZE) // 만약 BUF_SIZE만큼 읽지
        못했다면...
        {
            if (feof(ori) != 0) // 현재 파일이 EOF에 도
            달했는지, 즉 파일을 끝까지 읽었는지 확인한다.
            {
                fwrite(buffer, 1, count, cp);
                printf("Copy success!!\n");
                break;
            }
        }
```

```
        else // 만약 파일을 끝까지 읽은게
        아니라면, BUF_SIZE만큼 읽지 못한 것은 파일을
        읽는 과정에서 문제가 있었다는 이야기이므로...
        {
            printf("Copy FAIL!!\n");
            break;
        }
    }
    else
    {
        fwrite(buffer, 1, BUF_SIZE, cp);
    }
}

fclose(ori);
fclose(cp);
return 0;
}
```


•예제 6 실행 결과



•파일 입출력 및 구조체 과제

➤전화번호부 만들기

- 100명의 전화번호부를 만듭니다.
 - 이름, 전화번호, 부서를 저장할 수 있는 구조체를 선언합니다.
 - 구조체 배열을 선언하여 100명의 정보를 저장합니다.
- 만든 전화번호부를 .txt 파일로 출력합니다.