

# C Programming Assignment / Week 13

## 정리노트 #9

### 파일 입출력

- 기본적으로 파일에 데이터를 입출력하려면 파일을 열고(fopen), 처리하고(write,read), 닫는다(close)
- 파일 관련 함수는 헤더파일인 <stdio.h>에 존재한다.

```
#define _CRT_SECURE_NO_WARNINGS //Warning C4996 Error 방지코드(fopen_s로 썼는데도 실행이 안되서 넣었습니다)
#include <stdio.h>
#include <stdlib.h> //처음 보는 헤더파일 : 동적 메모리 관리, 의사 난수 생성등의 역할을 담당

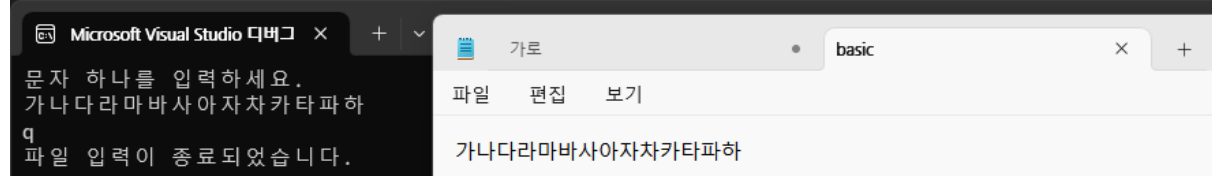
int main(void)
{
    FILE* fp; //파일 포인터 선언
    char ch;

    if ((fp = fopen("basic.txt", "w")) == NULL) //파일 포인터를 열기. (앞에 basic.txt는 파일 명이고 "w"는 파일 모드이다.
        //이때의 "w" = 쓰기 용도로 파일을 열며 새 파일을 만들, 만약 기존 파일이 존재한다면 그 내용을 다 지우고 새롭게 기록함
    {
        printf("파일이 열리지 않습니다.\n");
        exit(1);
    }

    printf("문자 하나를 입력하세요.\n");
    ch = getchar();

    while (ch != 'q') //만약 cmd에 q가 입력이 될 때 까지
    {
        fputc(ch, fp); //fp : 메모리에서 q 이전에 입력했던 데이터를 입력을 문자열 상태로 저장
        ch = getchar(); //ch : cmd에 작성한 내용들
    }

    printf("파일 입력이 종료되었습니다.\n");
    return 0;
}
```



약간의 부연 설명은 주석을 통해 작성했습니다.

+ 파일 포인터는 열고 닫기와 총 4가지의 입출력의 형태가 가능하다.(이중 저 코드에서 사용된건 fopen, fputc이다.)

```
//파일 포인터 열기 닫기 : fopen(), fclose() 이때, fclose()가 성공적으로 닫히면 0, 그렇지 않으면 EOF(Error Of File)출력
//입출력(4가지) : fgetc(), fputc() : 문자 입출력, fgets(), fputs() : 행 입출력, fread(), fwrite() : 블록 입출력, fscanf(), fprintf() : 형식화된 입출력
```

- 파일 열기, 닫기 및 파일 입출력 함수 정리

파일 열기 - fopen()

파일 닫기 - fclose()

문자 입출력 - fgetc(), fputc()

행 입출력 - fgets(), fputs()

블록 입출력 - fread(), fwrite()

형식화된 입출력 - fscanf(), fprintf()

fgetc, fputc 함수의 형식

> char fgetc(FILE\*) // FILE\* = 파일 포인터

> fputc(char, FILE\*) // char = 파일에 기록할 문자, FILE\* = 파일 포인터

fgets, fputs 함수의 형식

> char \*fgets(char \*, int, FILE\*); // char \* = 문자열이 저장된 배열을 가리키는 포인터, int = 문자열의 크기, FILE\* = 파일 포인터

> char \*fputs(char \*, FILE\*); // char \* = 문자열이 저장된 배열을 가리키는 포인터, FILE\* = 파일 포인터

fread()와 fwrite() 함수의 형식

fread()와 fwrite()는 둘 다 소괄호 안쪽이 같다.

> 함수명(void\*포인터 ①, 바이트 크기 ②, 블록 수 ③, FILE\* 포인터 ④);

```

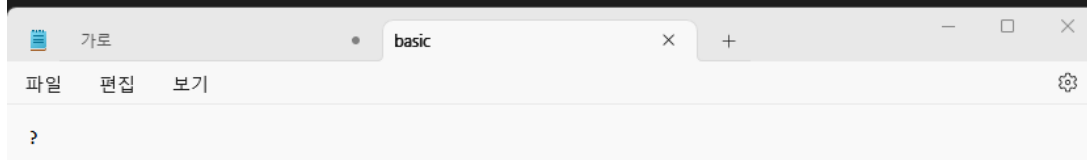
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    FILE* fp;
    int n = 4000;
    if ((fp = fopen("basic.txt", "wb")) == NULL) //이때 "wb" = 이전 파일을 읽기 용도로 열며 "w"와 같은 기능을 수행한다.
    {
        printf("파일을 열 수 없습니다.\n");
        exit(1);
    }

    fwrite(&n, sizeof(int), 1, fp);
    //이때, 함수가 오류 없이 정상적으로 출력 되면 블록 수를 반환하며 자료형의 바이트 크기 * 블록 수 만큼 파일에서 데이터를 기록함.

    fclose(fp); //FILE 포인터(fp에 작성된 텍스트) 닫기
    return 0;
}

```



1. 파일에서 읽어올(or 기록할) 자료를 가리키는 포인터(이 코드에선 n)
2. 읽어올(or 기록할) 자료형의 바이트 크기(이 코드에선 정수형의 크기)
3. 읽어올(or 기록할) 블록의 수(반복 횟수)(이 코드에선 1번)
4. FILE 포인터(이 코드에선 fp)

(약간의 부연 설명은 주석을 통해 작성했습니다.)

(fscanf, fprintf 파트)

fprintf(), fscanf() 함수의 형식

마찬가지로 소괄호 안쪽 구성은 같다.

> fprintf(FILE\* 포인터, 변환기호, 변수 목록); // 파일 포인터, printf() 함수에서 사용한 변환기호, 변수들

> fscanf(FILE\* 포인터, 변환기호, 변수 목록); // 파일 포인터, scanf() 함수에서 사용한 변환기호, 변수들

파일 임의 접근

파일 임의 접근에는 순차 접근과, 임의 접근이 있다.

순차 접근 = 파일의 처음이나 끝부터 데이터를 입출력하는 방식

임의 접근 = 임의의 위치에서부터 데이터를 입출력하는 방식, 파일 위치 지시자를 이용

파일 위치 지시자

(상수) SEEK\_SET > 파일의 처음 (값 = 0)

(상수) SEEK\_CUR > 파일의 현재 위치 (값 = 1)

(상수) SEEK\_END > 파일의 끝 (값 = 2)

fseek() 함수의 형식

> int \*fseek(FILE\* fp, long offset, int origin) // 파일 포인터, 바이트 단위로 origin부터 새로운 위치까지 떨어진 거리, 파일 위치 지시자

ex) fseek(fp, -50L, SEEK\_CUR): 파일의 현재 위치에서 50바이트 앞으로 이동

매크로

매크로(macro): 프로그래밍 할 때 반복적으로 나타나는 상수나 함수를 명령 하나로 새롭게 정의하는것.

(형식: #define 매크로 명, 대체할 값 (리스트명 수, 문자열 등))

Ex) 아래의 코드

```
#include <stdio.h>
#define PI 3.14 //PI(원주율) 값에 3.14값을 쓴 매크로를 정의

int main(void)
{
    int R; //반지름 정수형 R선언
    float cir; //원의 둘레인 실수형 cir 선언

    printf("원의 반지름을 입력하세요.\n");
    scanf_s("%d", &R);

    cir = PI * (2 * R); //매크로인 PI를 입력해서 원주를 구해 cir에 대입하는 장면
    printf("원의 둘레는 %.2f입니다.\n", cir);

    return 0;
}
```

Microsoft Visual Studio 디버그 × + ▾

원의 반지름을 입력하세요.  
4  
원의 둘레는 25.12입니다.

약간의 부연 설명은 주석으로 설명했습니다.

이번 페어 프로그래밍은 무엇으로 해볼까 생각하다가, 수업 때 풀은 연습문제 6을 활용한 연습문제 7을 하기로 했다.

수업시간에 한 6번은 1부터 100까지 누적하여 더한 것을 텍스트 파일을 생성하는 것이었지만, 7번은 이 텍스트 파일을 읽어들이고, 합이 300이상이 되지 않을 때 까지만 출력하는 것이다.

6번에서 txt 파일을 gg.txt로 설정하였다.

57 ~ 58줄

정민수: 파일 포인터와, 파일에서 읽은 값을 저장할 변수를 선언해주세요.

정재현: 파일 포인터 fp와, 정수형 변수 a1, a2, a3를 선언했습니다.

```

52 #define _CRT_SECURE_NO_WARNINGS
53 #include <stdio.h>
54 #include <stdlib.h>
55
56 int main() {
57     FILE* fp;
58     int a1, a2, a3;
59 }

```

정민수: 다음은 6번 문제에서 썼던 파일을 fopen을 이용하여 읽기 모드로 열어 주시고, 파일 열기 실패 시에는 에러 메시지를 띄워주세요.

정재현: 완료했습니다.

```

60 if ((fp = fopen("gg.txt", "r")) == NULL) {
61     printf("파일 열기 실패");
62     exit(1);
63 }

```

정민수: 그럼 이제 fscanf를 이용하여 변수 3개를 읽어들이고, 선언했던 정수형변수 3개에다가 넣어주세요. 그 다음엔, while문을 이용하여 반복하여 줍시다. while문이 지속되는 조건은 fscanf가 3개의 변수를 읽어오면 지속될 수 있도록, fscanf(~)==3을 넣어주세요. 그 다음, 합이 300 이상일 때는 출력하지 않는 방법을 구현해 보면 될 것 같아요.

정재현: if를 사용하여 a3, 즉 총합이 300이상이 되면 break 되게 코딩을 구현했습니다. 도스창에 딱 300이상이 되는 것은 출력하지 않고, 그 전까지만 출력 되었습니다.

```

51 #define _CRT_SECURE_NO_WARNINGS
52 #include <stdio.h>
53 #include <stdlib.h>
54
55 int main() {
56     FILE* fp;
57     int a1, a2, a3;
58
59     if ((fp = fopen("gg.txt", "r")) == NULL) {
60         printf("파일 열기 실패");
61         exit(1);
62     }
63
64     while (fscanf(fp, "%d %d %d\n", &a1, &a2, &a3) == 3) {
65         if (a3 >= 300) break;
66         printf("%d 부터 %d 까지의 합은 %d\n", a1, a2, a3);
67     }
68     fclose(fp);
69
70     return 0;
71 }

```

Microsoft Visual Studio 디버그 콘솔

1	부터	1	까지의	합은	1
1	부터	2	까지의	합은	3
1	부터	3	까지의	합은	6
1	부터	4	까지의	합은	10
1	부터	5	까지의	합은	15
1	부터	6	까지의	합은	21
1	부터	7	까지의	합은	28
1	부터	8	까지의	합은	36
1	부터	9	까지의	합은	45
1	부터	10	까지의	합은	55
1	부터	11	까지의	합은	66
1	부터	12	까지의	합은	78
1	부터	13	까지의	합은	91
1	부터	14	까지의	합은	105
1	부터	15	까지의	합은	120
1	부터	16	까지의	합은	136
1	부터	17	까지의	합은	153
1	부터	18	까지의	합은	171
1	부터	19	까지의	합은	190
1	부터	20	까지의	합은	210
1	부터	21	까지의	합은	231
1	부터	22	까지의	합은	253
1	부터	23	까지의	합은	276