파일입출력

1. 소스파일 나누기

   ▼ 파일 나누는 방법

   - 헤더 파일, 메인 함수 파일, 서브 함수 파일들로 파일을 나눠 작성한다.
   - 헤더 파일 내용
     1. 일반적으로 헤더 파일 이름은 서브 함수들이 있는 파일 이름을 사용한다.
        - 서브 함수가 들어있는 파일 이름이 sumAnother.c 라면 헤더 파일의 이름은 sumAnother.h 를 사용한다.
        - 헤더 파일에는 라이브러리 헤더, 사용자 정의 헤더, 함수의 원형 선언을 적는다.
        - 아래 예는 헤더파일 이름이 sumAnother.h 일 경우의 예이다.
     2. 라이브러리 헤더 : #include <stdio.h> 와 C가 제공하는 같은 라이브러리 헤더
     3. 사용자 정의 헤더 : #include "sumAnother.h" 와 같이 작성
     4. 함수의 원형 선언 : int addNumbers(int n);

   - 메인함수 파일: 사용자 정의 헤더 파일(#include sumAnother.h)과 메인 함수를 적는다.
   - 서브 함수 파일 : 사용자 정의 헤더 파일(#include sumAnother.h)과 서브 함수들을 적는다.

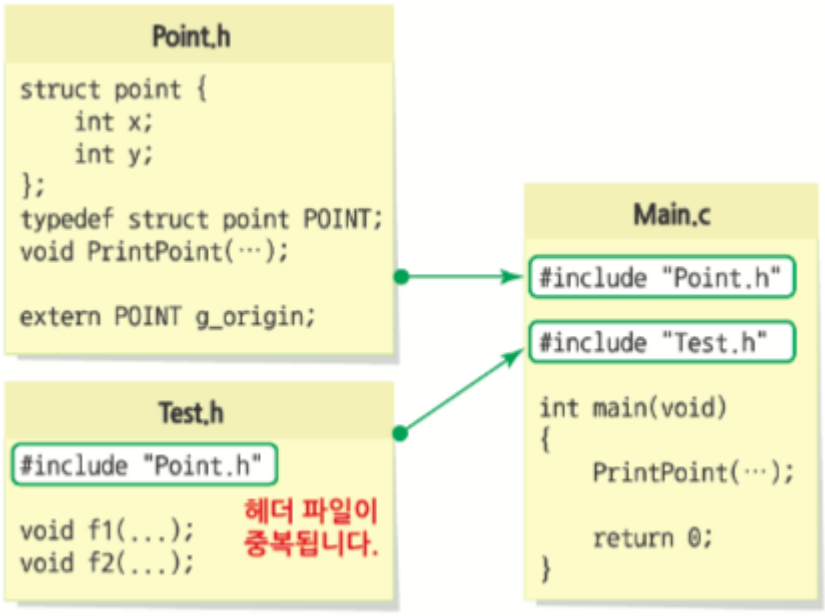
```
kth-ui-MacBook-Air:file_split danderlion$ more sumAnother.h
#include <stdio.h>
//#pragma once
int addNumbers(int n);

kth-ui-MacBook-Air:file_split danderlion$ more main.c
#include "sumAnother.h"

int main() {
    int num;
    printf("Enter a positive integer: ");
    scanf("%d", &num);
    printf("Sum = %d", addNumbers(num));
    return 0;
}
kth-ui-MacBook-Air:file_split danderlion$ more sumAnother.c
#include "sumAnother.h"

int addNumbers(int n) {
        int i,sum=0;

        for (i=0;i<n+1;i++){
                sum += i;
        }
        return sum;
}
kth-ui-MacBook-Air:file_split danderlion$
```

▼ 헤더 파일에 넣을 수 있는 내용

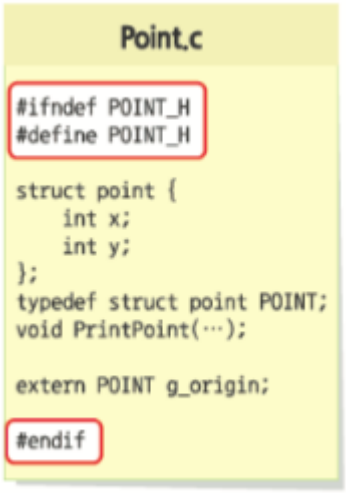
- 함수 선언
- 구조체 정의
- typedef 정의
- 전역 변수의 extern 선언
- 매크로 정의

▼ 헤더 파일 중복

- 헤더 파일에 구조체, 매크로, typedef 가 있을 경우 헤더파일이 중복 되면 컴파일 에러가 난다.



- #ifndef , #define, #endif 를 사용하여 컴파일 과정에 중복 포함 되는 것을 막을 수 있다.
- #ifndef = if not defined 의 뜻으로, #ifndef 다음에 나오는 매크로가 정의 되어 있지 않은 경우에만 #ifnde와 #endif 사이의 문장이 컴파일 된다.
- 그림에서 #ifndef 다음에 나오는 매크로는 POINT_H 이다.



2. main 함수의 원형

▼ 사용 가능한 main 함수의 원형

```
void main(void);
int main(void);
int main(int argc, char *argv[]);
```

▼ int main(int argc, char *argv[]);

- argc 는 명령행 인자의 개수이다.
- argv 는 명령행에 쓰여져 있는 문자열을 배열로 받아 들인다.

```
./a.out data.txt  // argc == 2 가 된다.

argv[0] = ./a.out
argv[1] = data.txt
```

3. printf

▼ 형식 문자열 구조

```
%[flags][witdth][.precision][{h | l | L}]type
```

▼ flags

flags	의미	디폴트 값	사용 예	출력 결과
-	전체 폭에 대하여 왼쪽으로 정렬한다.	오른쪽으로 정렬한다.	printf("%-10d", 123);	123□□□□□□□
+	부호 있는 값에 대하여 +/-를 출력한다.	음수에 대해서만 -를 출력한다.	printf("%+d", 123);	+123
빈칸	양수면 부호 자리에 빈칸을 출력한다.	빈칸을 출력하지 않는다.	printf("% d", 123); printf("% d", -123);	□123 -123
0	정수의 폭에 맞춰서 0으로 채운다.	0으로 채우지 않는다.	printf("%010d", 123);	0000000123
#	o, x, X와 함께 사용되면 0, 0x, 0X를 함께 출력한다.	0, 0x, 0X를 출력하지 않는다.	printf("%#x", 123);	0x7b

▼ width, precision

%10.4d 의 경우 10 이 width 로 전체 폭이10 자리, .4 는 정밀도를 나타내는 것으로 정수에서는 정수를 4자리로(0012), 실수에서는 소수점 아래 4자리까지 표시

▼ h | l | L

h : short 형 출력

l : long 형 출력

L : long double 형 출력

▼ type

유형	type	의미	사용 예	출력 결과
정수	c	문자를 출력한다.	printf("%c", 'A');	A
	d	10진수로 출력한다.	printf("%d", 123);	123
	i	10진수로 출력한다.	printf("%i", 123);	123
	o	8진수로 출력한다.	printf("%o", 123);	173
	u	부호 없는 10진수 정수를 출력한다.	printf("%u", 123);	123
	x	abcdef를 이용해서 16진수 정수로 출력한다.	printf("%x", 123);	7b
	X	ABCDEF를 이용해서 16진수 정수로 출력한다.	printf("%X", 123);	7B
실수	e	지수 표기로 출력한다.	printf("%e", 12.3456);	1.234560e+001
	E	지수 표기로 출력한다.	printf("%E", 12.3456);	1.234560E+001
	f	소수 표기로 출력한다.	printf("%f", 12.3456);	12.345600
	g	지수와 소수 표기 중 더 간단한 형식으로 출력한다.	printf("%g", 12.3456);	12.3456
	G	지수와 소수 표기 중 더 간단한 형식으로 출력한다.	printf("%G", 12.3456);	12.3456
포인터	p	16진수로 주소를 출력한다.	printf("%p", "abc");	00426054
문자열	s	문자열을 출력한다.	printf("%s", "abc");	abc

4. scanf

▼ 형식 문자열 구조

```
%[*][width][{h|l|L}]type
```

▼ *

"%d %*d %d" 를 사용하면 두 번째로 입력되는 정수를 무시한다.

▼ width = 입력받을 값의 폭 지정

```
"%2d" 로 12345 를 입력받으면 12 가 입력 된다.
```

▼ type

```
%c 문자열
%d, %i, %u, %o, %x, %X 정수
%e, %E, %f, %g, %G 실수 (double -> %lf)
```

5. 파일열기

▼ fopen 함수 사용 (FILE 구조체 포인터)

```
FILE *fopen(const char *filename, const char *mode);
```

▼ 파일 열기 모드

- 파일 열기 모드 : 읽기 전용(r), 쓰기 (w), 내용추가 (a), 바이너리 입출력(b)

```
FILE *fp = NULL;
fp = fopen("data.data", "rb");
```

6. 파일닫기

▼ fclose 함수로 파일을 닫는다.

```
int fclose(FILE *stream);

예> fclose(pointer_of_file_name);
```

7. 파일 읽기 / 쓰기

▼ 한글자 읽기 / 쓰기

- fgetc , fputc 함수 이용

```
FILE *fp1, *fp2;
char ch;

fp1 = fopen(argv[1], "r");
fp2 = fopen(argv[2], "w");
```

```
ch=fgetc(fp1);
fputc(ch, fp2);

fclose(fp1);
fclose(fp2);
```

▼ 문자열 읽기 / 쓰기

- fgets, fputs 사용

```
FILE *fp1, *fp2;
char buffer[MAX_BUF];  // 읽어들인 문자열을 받아 들일 배열 선언

fp1 = fopen(argv[1], "r");
fp2 = fopen(argv[2], "w");

fgets(buffer, MAX_BUF, fp1);
fputs(buffer, fp2);

fclose(fp1);
fclose(fp2);
```

▼ 변수단위로 읽기 / 쓰기

- fscanf: 파일로 부터 변수 단위로 입력을 받는다.
- 변수간의 구분은 공백, 탭, 개행문자로 한다.
- fprintf: 변수 단위로 파일에 저장 한다.
- fscanf 와 fprintf 의 사용법은 scanf, printf 와 사용법이 같다. (ex1301.c)