C 프로그래밍 기법 향상

김경민

재귀호출(recursive call)

- **함수 안에서 함수 자기자신을 호출하는 방식**
  - 재귀호출은 일반적인 상황에서는 잘 사용하지 않지만 알고리즘을 구현할 때 매우 유용
  - **재귀호출을 사용하려면 반드시 다음과 같이 종료 조건**

```c
#include <stdio.h>

void show(int n) {
    if ( n < 1 ) return ;

    printf("%d\n", n) ;
    show(n-1) ;
}

int main() {
    int n = 5;

    show(n) ;
}
```

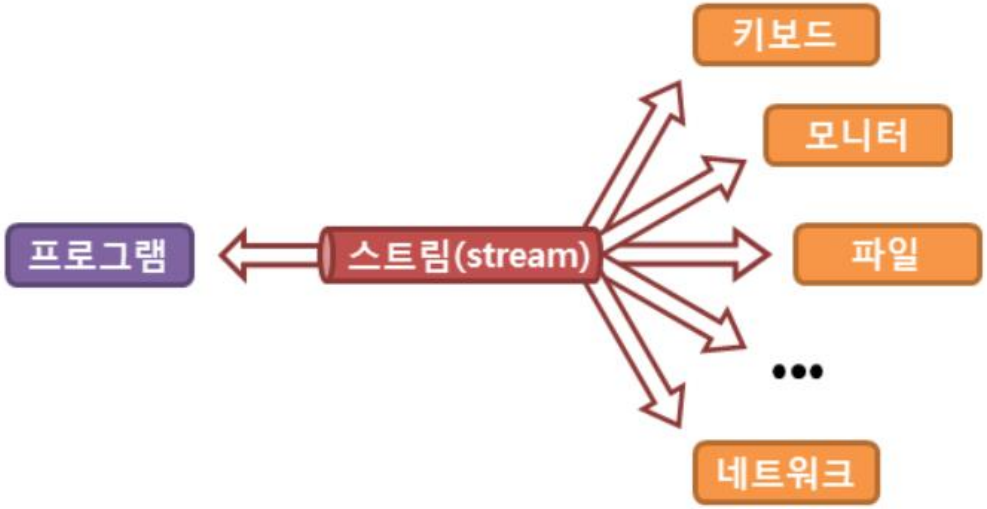
```c
#include <stdio.h>

void show(int n) {
    if ( n < 1 ) return ;

    show(n-1) ;
    printf("%d\n", n) ;
}

int main() {
    int n = 5;

    show(n) ;
}
```

해결문제

- 1에서 10까지 합계를 구하시오.
  - 재귀함수를 이용
- n!를 구하시오.
  - 재귀함수를 이용
- 10진수를 이진수로 변환하시오.
  - 재귀함수를 이용

스트림(stream)

- 실제의 입력이나 출력이 표현된 데이터의 이상화된 흐름
- 운영체제에 의해 생성되는 가상의 연결 고리
- C 프로그램은 파일이나 콘솔의 입출력을 직접 다루지 않고, 스트림(stream)이라는 것을 통함



파일 입출력

- 파일(file)
  - 의미 있는 정보를 담고 있으며, 이름을 가지고 있는 저장 장치상의 논리적인 단위를 의미
  - C언어에서는 이러한 파일을 바이트별로 따로 읽을 수 있는 연속적인 바이트의 집합으로 취급
- 파일의 종류
  - 텍스트 파일(text file)
    - 사람이 알아볼 수 있는 문자열로 이루어진 파일
  - 바이너리 파일(binary file)
    - 데이터의 저장과 처리를 목적으로 0과 1의 이진 형식으로 인코딩된 파일

파일에 대한 입출력 동작

1. 파일과의 스트림 생성
   - FILE *fopen(const char *filename, const char *mode);
   - mode를 통해 대상 파일이 Text File 인지 Binary File인지 지정
2. FILE 구조체 변수의 포인터를 이용한 작업 진행
   - Text File : fgetc(), fputc(), fgets(), fputs(), fscanf(), fprintf()
   - Binary File : fread(), fwrite()
   - feof 함수
     - 현재 파일 포인터의 위치가 파일의 끝이면 1, 파일의 끝이 아니면 0을 반환하므로 -1인 EOF와는 상관이 없음.
     - EOF는 scanf, fscanf 등의 함수에서 값을 읽을 수 없는 상태일 때 반환
3. 파일과의 스트림 종결
   - `int fclose( FILE *stream );`

파일과의 스트림 생성

- **FILE *fopen(const char *filename, const char *mode);**
  - filename : Open할 파일에 대한 경로명을 담고 있는 문자열
  - mode : 파일은 어떤 모드로 Open할 것인지를 나타내는 문자열

mode		의미	파일이 없으면	파일이 있으면
Text	Binary			
"r"	"rb"	읽기 전용 (read)	NULL 반환	정상 동작
"w"	"wb"	쓰기 전용 (write)	새로 생성	기존 내용 삭제
"a"	"ab"	추가 쓰기 (append)	새로 생성	기존 내용 뒤에 추가
"r+"	"rb+"	읽기와 쓰기	NULL 반환	정상 동작
"w+"	"wb+"	읽기와 쓰기	새로 생성	기존 내용 삭제
"a+"	"ab+"	추가를 위한 읽기와 쓰기	새로 생성	기존 내용 뒤에 추가

텍스트 파일 입출력

```
#include <stdio.h>
typedef struct {
        int left , top , right , bottom;
} RECT;

int main() {
        FILE *fp;
        RECT rect;
        int no = 0;

        fp = fopen("test.txt" , "w");
        while(no != 3) {
                scanf("%d %d %d %d", &rect.left , &rect.top , &rect.right , &rect.bottom) ;
                fprintf(fp, "%d %d %d %d₩n", rect.left , rect.top , rect.right , rect.bottom);
                no++;
        }
        fclose(fp) ;

        fp = fopen("test.txt" , "r");
        while(fscanf(fp, "%d %d %d %d", &rect.left , &rect.top , &rect.right , &rect.bottom) != EOF) {
                printf("%d %d %d %d₩n", rect.left , rect.top , rect.right , rect.bottom);
        }
        fclose(fp) ;
        return 0;
}
```

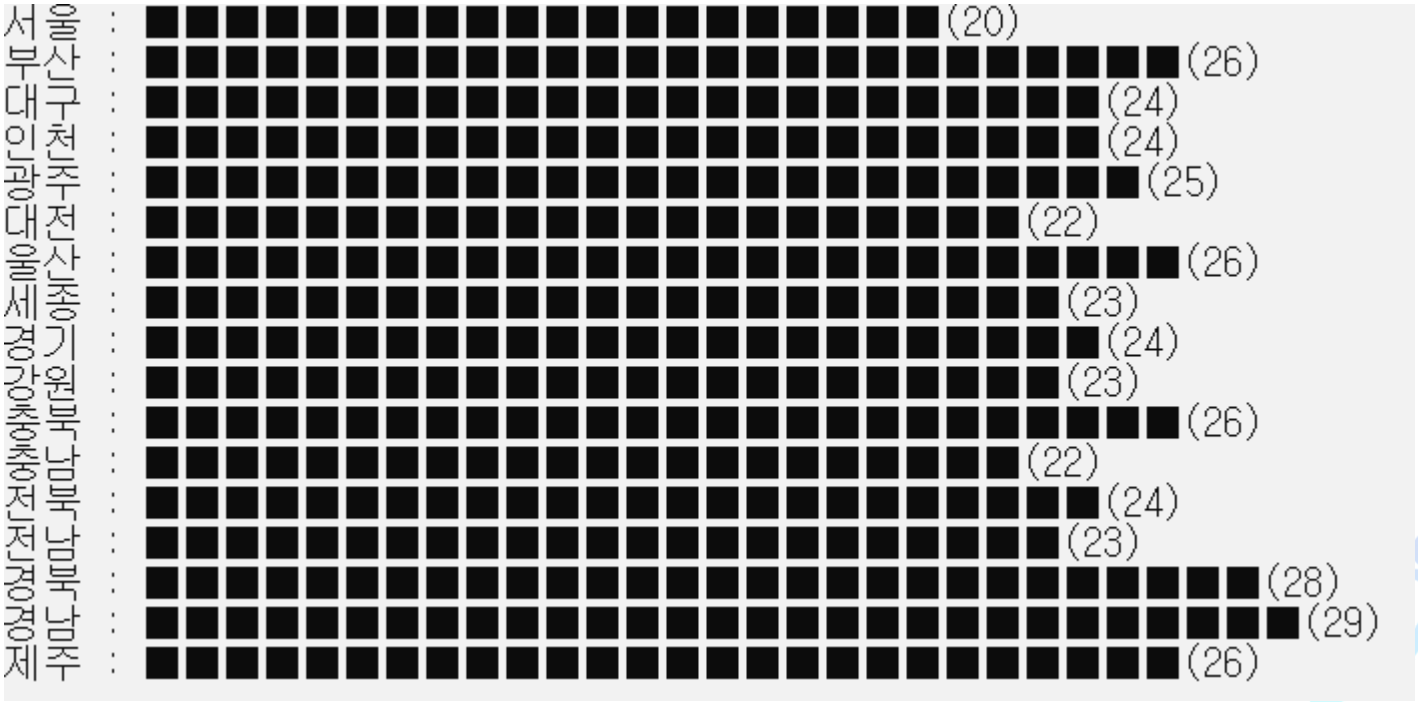
바이너리 파일 입출력

```
#include <stdio.h>
typedef struct {
        int left , top , right , bottom;
} RECT;

int main() {
        FILE *fp;
        RECT rect;
        int no = 0;

        fp = fopen("test.txt" , "wb");
        while(no != 3) {
                scanf("%d %d %d %d", &rect.left , &rect.top , &rect.right , &rect.bottom) ;
                fwrite(&rect , sizeof(RECT) , 1 , fp);
                no++;
        }
        fclose(fp) ;

        fp = fopen("test.txt" , "rb");

        while(1) {
                fread(&rect , sizeof(RECT) , 1 , fp);
                if (feof(fp)) break;
                printf("%d %d %d %d₩n", rect.left , rect.top , rect.right , rect.bottom);
        }

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

https://hexed.it/ 확인

해결문제

- 도별미세먼지.txt 파일을 읽어서 화면에 표시하시오.



해결문제

- N개의 좌표를 입력 받아서 거리를 구하시오.

```
5
0 0
3 4
3 8
5 8
0 0
p(0 ,0) p(3 ,4) distance = 5.0000
p(3 ,4) p(3 ,8) distance = 4.0000
p(3 ,8) p(5 ,8) distance = 2.0000
p(5 ,8) p(0 ,0) distance = 9.4340
```

선행처리(preprocess)

- 실행 파일을 생성하는 과정에서 소스 파일 내에 존재하는 선행처리 지시문을 처리하는 작업
  - 선행처리 작업은 컴파일하기 전 선행처리기(preprocessor)에 의해 먼저 처리
  - 코드를 생성하는 것이 아니라, 컴파일러가 컴파일하기 좋도록 소스를 재구성해 주는 역할

- 특징
  - 선행처리 문자(#)로 시작
  - 맨 뒤에 세미콜론(;)을 붙이지 않음
  - 선행처리문이 위치한 곳에서부터 파일의 끝까지만 영향

선행처리 종류

- **#include**
  - 외부에 선언된 함수나 상수 등을 사용하기 위해서 헤더 파일을 현재 파일에 포함
  - C언어에서 제공하는 표준 헤더 파일을 포함할 때에는 보통 꺾쇠괄호(<>)를 사용
    - #include <stdio.h>
  - 사용자가 직접 작성한 헤더 파일을 포함할 때에는 보통 큰따옴표("")를 사용
    - #include "myStdio.h"

선행처리 종류

- **#define**
  - 함수나 상수를 단순화 해주는 매크로를 정의할 때 사용
    - 매크로는 함수나 상수에 이름을 붙임
  - 매크로 확장(macro expansion)
    - #define 선행처리 지시문의 식별자(identifier)를 단순히 대체 리스트(replacement-list)로 치환

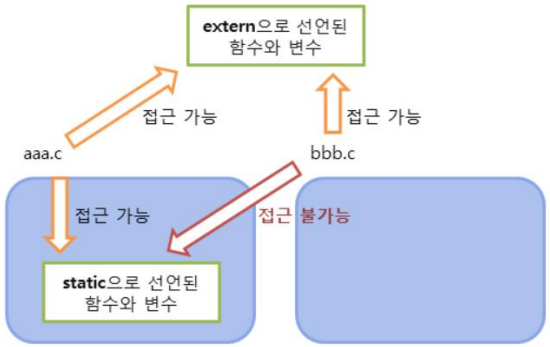
매크로 함수

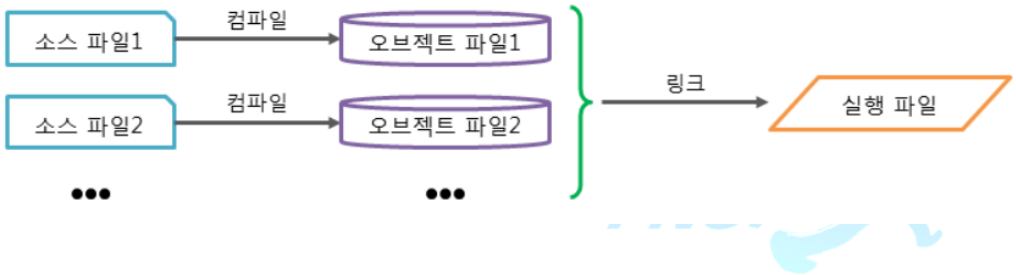
- **#define 선행처리 지시문에 인수로 함수의 정의를 전달함으로써, 함수처럼 동작**
  - 일반 함수와는 달리 단순 치환만 수행

```c
#include <stdio.h>
#define ADD(x, y) x+y
#define SQRT(x) x*x
int main() {
    int a = 10, b = 20 ;

    printf("%d + %d = %d\n", a,b,ADD(a,b)) ;
    //단순 치환 : 10+2*10+2
    printf("%d^2 = %d\n", a+2,SQRT(a+2)) ;
    return 0;
}
```

분할 컴파일

- **하나의 실행 파일을 만들기 위해서 소스 파일을 여러 개로 나누어 개발하는 방식**
  - extern 키워드
    - 외부 파일에서 선언된 전역 변수를 참조하기 위해서는 파일 내에서 extern 키워드를 사용해 다시 한 번 변수를 선언
  - static 키워드
    - 변수의 접근 영역을 해당 파일로만 한정시키고자 할 때는 static 키워드를 사용하여 선언





조건부 컴파일(conditional compile)

- 지정한 조건에 따라 코드의 일정 부분을 컴파일 할지 안 할지를 지정
- 조건부 컴파일 지시자(conditional compile directive)
  - #if , #ifdef, #ifndef

```
#include <stdio.h>
#define COND 2

int main(void)
{
    #if COND == 1
        puts("매크로 상수 COND가 선언되어 있으며 그 값은 1입니다.");
    #elif COND == 2
        puts("매크로 상수 COND가 선언되어 있으며 그 값은 2입니다.");
    #elif COND == 3
        puts("매크로 상수 COND가 선언되어 있으며 그 값은 3입니다.");
    #endif
    return 0;
}
```