Project 1

영화관

Project Constructed by 조규상

연무고등학교

I. 연습

구조체, 포인터, 메모리 동적할당, 자료구조 linked list

II. 프로젝트 설명

작은 영화관의 주인이 된 우리는 C언어를 이용해 영화관을 관리하는 프로그램을 만드려고 한다. 여러분들에겐 이미 기초적인 building이 완료된 skeleton code가 제공된다. Code와 아래의 설명을 차근차근 읽어보고, 뒤의 [III 목표]를 풀어내보자.

1. 이론적 배경

다음의 내용들은 여러분들이 프로젝트를 진행하기 위하여 반드시 알아야만 하는 이론들이다. 일부 내용은 소프트웨어 중급과정을 넘어서지만, 난이도는 크게 어렵지 않기 때문에 아래의 설명을 자세히 읽어보기를 바란다.

1) Typedef : 프로그램의 간결성을 위해 사용되는 typedef이다.
구조체에서 많이 쓰이며, 그 자료형의 별명이라고 생각하면 된다.

```
typedef struct movie{
  int mov_no;
  char title[50];
  int year;
  GENRE g;
  struct movie* next;
} MOV_T;
```

struct movie의 자료형을 가지는 변수를 선언할 때 우리는 struct movie m; 과 같이 선언한다.

이렇게 typedef를 정의해주게되면 이후 MOV_T를 struct movie 대신 사용할 수 있다. 즉 MOV_T m; 와 같이 선언할 수 있다는 것이다. 물론 struct movie로 써주어도 아무 문제없지만, MOV_T라고 써주는 것이 조금 더 짧기 때문에 많은 프로그램에서는 이런 방식을 택하고 있다.

```
typedef int I;
I k=5;
```

```
int k=5;
```

예를 들어, 위의 두 개는 똑같은 코드이며, 좌측처럼 쓰더라도 아무 문제없이 프로그램

이 잘 돌아간다.

2) Enumeration , 열거형

```
typedef enum genre{
        G_NONE=0,
        G_ACTION = 1,
        G_ADVENTURE=2,
        G_THRILLER=4,
        G_HORROR=8,
        G_COMEDY=16
} GENRE;
```

열거형은 프로그램에 대한 이해도를 향상시키기 위해 사용한다.
어떤 데이터가 정해진 몇 개의 값 중에 하나를 가지게 할 경우 사용된다.
아주 간단한 예시를 들어보자.

```
typedef enum gender{
        G_MALE=0,
        G_FEMALE=1
} GENDER;
GENDER m = G_MALE;
```

위와 같은 코드가 있다고 가정하자. 이 때, G_MALE 이나 G_FEMALE은 변수명이 아니며, 각각 0 과 1을 대변하는 한 기호라고 생각하면 된다.
위와 같은 예시일 경우, 5번째 줄에서 m에 0이 들어가게된다. 하지만 0 대신 G_MALE이라는 표현을 사용하기 때문에 가독성이 뛰어난 코드를 작성하기 용이하다.

3) Linked list

리스트라는 자료구조이다. 아주 기본적인 구조체를 생각해보자.

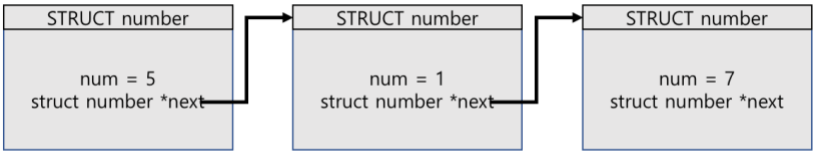
```
struct number{
 int num;
}
```

위와 같은 구조를 가지는 구조체가 하나만 존재한다면, 관리하기 쉽다.
하지만, 위의 구조를 가지는 연속적인 구조체들이 여러 개가 있을 경우 관리하기가 급격히 어려워진다. 이 구조체 하나를 element라고 하자. 총 element의 개수가 정해져 있다면 위와 같은 구조체들의 배열의 형태로 둘 수 있지만, 총 element의 개수에 대해 알지 못할 경우 우리는 linked list를 이용한다.
다음과 같이 바로 다음 struct number의 주소를 원 구조체 안에 포함시킬 수 있다.

즉, 현재 element의 성분에 다음 element의 주소를 포함시킨다는 것이다. 이렇게 할 경우 현재 element에서 다음 element에 접근하는 것이 가능해진다.

```
struct number{
 int num;
 struct number* next;
}
```

이런 형태를 linked list라고 하며, 아래와 같이 도식화할 수 있다.



자세한 내용은 소프트웨어 – 고급과정에서 다루게된다.

2. Movie Theater

본 프로젝트는 크게 **(1) 영화들을 관리하는 부분**과 **(2) 영화관을 관리하는 부분** 두 개로 나뉜다.

(1) 영화

입력으로 영화 정보가 주어진다. 이 때 이 영화들의 정보를 linked list의 구조에 정확하게 저장해야 하며, 이 list를 제어할 수 있는 간단한 function들을 만든다.

(2) 영화관

(1) 에서 만든 영화 정보들을 바탕으로, 총 5관으로 구성된 영화관을 관리하는 프로그램을 만든다.

Skeleton Code를 살펴보자.

1) struct [movie]

struct movie 는 아래와 같이 선언되어있다.

```
typedef struct movie{
 int mov_no;
 char title[50];
 int year;
 GENRE g;
 struct movie* next;
} MOV_T;
```

mov_no는 영화의 고유 번호를 의미한다. 이 번호를 통해 영화관에서 상영할 영화를 확인

한다. Title은 그 영화의 제목, year는 영화 개봉 년도, genre에는 장르가 저장된다. 또한 linked_list의 구조를 가지기 위해 struct movie pointer next를 하나 가진다. 현재 skeleton code에는 title과 year, next가 이미 저장되도록 되어 있으며, 이 list에서 장르 정보를 변경하는 것은 여러분들의 임무(Q3)이다.

2) struct [mov_header]

struct mov_header는 struct movie로 구성된 linked_list의 정보를 담아둔다.

```
typedef struct mov_header{
 MOV_T* first;
 int list_length;
} MOV_H;
```

위와 같이 구성되며, 해당 리스트에 포함되어있는 첫 번째 movie element에 대한 주소정보를 포함하고 있다. 또한 list_length도 저장되어있고, 이 정보를 update하는 것도 여러분들의 임무(Q2)이다.

movie_list 한 개가 전역변수로 선언되어 있으며, 이 것만 이용하면 된다.

3) Input files

프로젝트 파일에 포함되어 있는 movie.txt와 theater.txt 파일을 확인해보자.

입력으로 이 두 파일을 읽어올 것이며, movie.txt에 대한 입력 처리는 file_input_movie()에서 이뤄진다. 그 후 new_movie()함수에서 새로운 movie struct를 생성한 후 movie_list에 추가한다.

나머지 정보는 코드를 천천히 읽으며 이해하길 바란다.

III. 목표

PART 1. 영화

1. 프로그래밍시 가장 중요한 부분 중 하나는 debugging 이다. movie_list에 영화들이 잘 삽입되고 있는지를 확인할 수 있는 함수인 print_all_movie()를 완성하시오.
2. mov_header의 struct를 가지는 movie_list 에는 list_length라는 element가 포함되어 있다. 이 list_length에 movie_list에 포함되어 있는 movie의 개수가 항상 저장되도록 프로그래밍하시오. (Hint: put_movie_on_list()를 바꾸시오.)
3. file_input_movie() 함수를 자세히 보자. 이 함수에서는 같은 폴더에 존재하는 movie.txt 에서 영화 정보를 가져온다. movie.txt의 데이터 입력 형식을 확인해보자.

   이 함수에서는 모든 영화에 일괄적으로 G_NONE, '장르 없음'을 넣고 있다. 장르 정보도 struct movie에 저장되도록 프로그래밍하시오. 장르가 Action, Adventure, Thriller, Horror, Comedy가 아닐 경우 그대로 G_NONE을 넣어주면 되며, (Hint: 수업시간에 배웠던 strtok, strcpy, strcmp와 같은 문자열 처리 함수를 이용하시오.) 만약 두 개 이상의 장르가 겹치는 경우 해당하는 아무 장르나 넣어줘도 됨.

Part 2. 영화관

4. search_list_by_no 함수는, num이라는 정수형 변수를 인자로 받아 그 num을 mov_no로 가지는 movie를 찾아 return 해주는 함수이다. 이 함수를 완성하시오.
5. 1에서 작성한 함수를 이용해, 영화관의 상영 정보를 저장하고 있는 movie_theater 배열을 확인해보자. movie_theater의 5개의 관에는 현재 첫 번째 영화만 상영하고 있다. 다음의 규칙에 맞게 프로그램을 다시 짜보자.
    1) 모든 상영관은 서로 다른 영화를 상영한다.
    2) Theater.txt에서 입력되는 숫자들에 해당하는 mov_no를 가지는 영화가 순서대로 1관부터 배치된다.
    3) Theater.txt에 입력되는 숫자의 수는 5개 이상이다.
    4) 이미 상영되고 있는 영화가 입력으로 들어온다면, 상영정보에 변화를 주지 않는다.
    5) 예를 들어, theater.txt 에서 10 5 81 2 45 22 11 81의 값이 입력된다면, 다음과 같이 배치하면 된다.

	1관	2관	3관	4관	5관
10	10	1	1	1	1
5	10	5	1	1	1
81	10	5	81	1	1
2	10	5	81	2	1
45	10	5	81	2	45
22	22	5	81	2	45
11	22	11	81	2	45
81	22	11	81	2	45

그러므로 총 22, 11, 81, 2, 45의 영화가 상영되어야한다.
print_movie_theater()함수를 이용해서 확인하라.