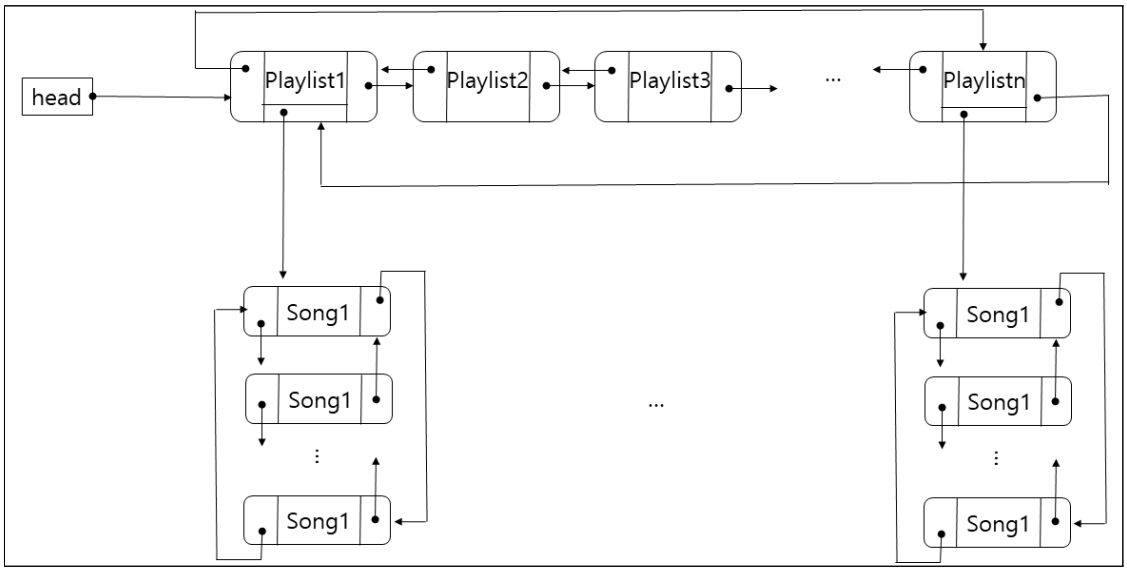
자료구조 실습과제 09 - 이중 원형 연결리스트 활용

- 솔루션 및 프로젝트 명칭 : Proj_09_이름이니셜

- 제출방법 : 아래 문제를 해결하기 위한 프로그램을 구현한 후 컴파일 및 실행한 후, 오류가 없으면 메뉴에서 **솔루션 정리를 수행한 후 윈도우 탐색기에서 솔루션 폴더를 찾아 압축하여** E-class에 올림

문제 1) 이중 원형 연결리스트를 이용한 음악 플레이리스트 관리하는 프로그램 작성

아래 그림과 같이 음악 플레이리스트를 관리하는 프로그램을 작성하시오. 플레이리스트는 노래들을 모아 놓은 하나의 그룹이며, 플레이리스트에는 사용자가 그 그룹에 저장해 놓은 노래들이 들어 있다. 사용자는 플레이리스트와 플레이리스트 내에 들어 있는 노래들을 관리하는 기능을 사용할 수 있다.



플레이리스트를 관리하는 메뉴와 노래를 관리하는 메뉴는 아래와 같다. 노래를 관리하는 메뉴는 특정 플레이리스트가 선택된 상태에서 s(song menu)가 선택되면 실행되는 메뉴이다.

```
void print_playlist_menu(void)
{
        printf("\n\n---- Playlist Menu ----     \n");
        printf(" a : print all playlist     \n");
        printf(" c : print current playlist\n");
        printf(" n : new playlist           \n");
        printf(" d : delete playlist        \n");
        printf(" > : next playlist          \n");
```

```
        printf(" < : previous playlist      \n");
        printf(" s : song menu              \n");
        printf(" q : quit                   \n");
        return;
}
void print_song_menu(void)
{
        printf("\n\n---- Song Menu ----     \n");
        printf(" a : print all songs     \n");
        printf(" c : print current song\n");
        printf(" n : add new song in playlist    \n");
        printf(" d : delete current song         \n");
        printf(" > : next song           \n");
        printf(" < : previous song       \n");
        printf(" b : quit                   \n");
        return;
}
```

위 그림과 같이 구성되는 음악 플레이리스트 관리 프로그램을 이중 연결리스트를 이용하여 구현한다. 플레이리스트의 한 노드는 플레이리스트명과 자기 앞쪽과 뒤쪽의 플레이리스트 노드를 가리키는 포인터와 노래 노드를 가리키는 포인터를 포함한다.
플레이리스트를 위한 명령어는 다음과 같이 수행된다.

- 프로그램을 실행하면 head는 NULL 값을 가지며, n 명령어를 이용하여 새로운 플레이리스트를 추가할 수 있다. 새로운 플레이리스트를 추가하면 현재 선택된 플레이리스트로 추가한 플레이리스트로 설정한다.
- 선택된 플레이리스트는 <를 통해 앞쪽, >를 통해 뒤쪽으로 이동할 수 있다.
- d 명령어를 통해 현재 선택된 플레이리스트를 삭제한다. 이 경우 선택된 플레이리스트는 삭제된 플레이리스트 다음의 리스트로 설정한다. 만약 마지막 플레이리스트를 삭제할 경우 선택된 플레이리스트는 NULL이 된다.
- a 명령어는 실행 예와 같이 저장되어 있는 플레이리스트를 모두 출력하며, 선택된 플레이리스트는 [* *]로 구분된다.
- c 명령어는 현재 선택된 플레이리스트를 출력한다.
- s 명령어는 현재 선택된 플레이리스트가 있을 경우 노래를 관리하기 위한 메뉴로 진입한다.
- q 명령어는 프로그램을 종료한다.

노래를 관리하는 방법은 이중 연결리스트로 구현되며, 노래를 위한 노드는 노래명과 자기 앞/뒤 노드를 가리키는 포인터로 구성된다.

- 선택된 플레이리스트에서 노래 관리 메뉴로 처음 들어올 경우 플레이리스트의 노래 노드를 가리키는 포인터는 NULL 값을 가지며, n 명령어를 이용하여 새로운 노래를 추가할 수 있다. 새로운 노래를 추가하면 현재 선택된 노래로 추가한 노래를 설정한다. 만약 선택된 플레이리스트 내에 노래가 존재한다면 플레이리스트 내의 첫 번째 노래가 현재 선택된 노래

로 설정한다.

- 선택된 노래를 <를 통해 앞쪽, >를 통해 뒤쪽으로 이동할 수 있다.
- d 명령어를 통해 현재 선택된 노래를 삭제한다. 이 경우 선택된 노래는 삭제된 노래 다음의 리스트로 설정한다. 만약 마지막 노래를 삭제할 경우 선택된 노래는 NULL이 된다.
- a 명령어는 실행 예와 같이 저장되어 있는 노래를 모두 출력하며, 선택된 노래는 [* *]로 구분된다.
- c 명령어는 현재 선택된 노래를 출력한다.
- b 명령어는 노래관리 메뉴에서 플레이리스트 관리 메뉴로 되돌아 간다.

아래 메인 함수의 구현 예를 참고하여 프로그램을 작성하시오.

실행 예

```
int main(void)
{
	DPlaylistNode* head = NULL;
	element buf;

	DPlaylistNode *currentPlaylist=NULL;
	DSongNode *currentSong=NULL;
	char menu, sub_menu;

	do {
		//printf("메뉴 선택");
		print_playlist_menu();
		printf("Select Menu : ");
		scanf(" %c", &menu);
		printf("-------------------------\n");
		printf("Selected playlist menu : %c \n", menu);

		switch (menu) {
		case 'a':
			print_aPlaylist(head, currentPlaylist);
			break;
		case 'c':
			printf("Current playlist : %s\n", currentPlaylist->pName);
			break;
		case 'n':
			printf("Playlist name : ");
			scanf("%s", buf);
			insert_nPlaylist(&head, buf, &currentPlaylist);
			print_aPlaylist(head, currentPlaylist);
			break;
		case 'd':
			delete_cPlaylist(&head, &currentPlaylist);
			print_aPlaylist(head, currentPlaylist);
			break;
		case '<':
			change_cPlaylist(&currentPlaylist, 2);
			print_aPlaylist(head, currentPlaylist);
			break;
		case '>':
			change_cPlaylist(&currentPlaylist, 1);
```

```
                        print_aPlaylist(head, currentPlaylist);
                        break;
                case 's' :
                        currentSong = currentPlaylist->slink;
                        do {
                                printf("Cur PL : %s, Cur Song : %s \n",
                                        currentPlaylist->pName,
currentSong->sName);
                                print_song_menu();
                                printf("Select Menu : ");
                                scanf(" %c", &sub_menu);
                                printf("Selected song menu : %c \n", sub_menu);


                                switch (sub_menu) {
                                case 'a':
                                        print_aSong(currentPlaylist, currentSong);
                                        break;
                                case 'c':
                                        printf("current song : %s\n",
currentSong->sName);
                                        break;
                                case 'n':
                                        printf("Song name : ");
                                        scanf("%s", buf);
                                        insert_nSongList(currentPlaylist, buf,
&currentSong);
                                        print_aSong(currentPlaylist, currentSong);
                                        break;
                                case 'd':
                                        delete_cSong(currentPlaylist, &currentSong);
                                        print_aSong(currentPlaylist, currentSong);
                                        break;
                                case '<':
                                        change_cSong(&currentSong, 2);
                                        print_aSong(currentPlaylist, currentSong);
                                        break;
                                case '>':
                                        change_cSong(&currentSong, 1);
                                        print_aSong(currentPlaylist, currentSong);
                                        break;
                                }
                        }while (sub_menu != 'b');
                        break;
                }
        } while (menu != 'q');

        return 0;
}
```

실행 예 :

첨부 파일의 실행 예 참고하세요.