|  |
| --- |
| 2020년 1학기 |
| 프로그래밍 언어 레포트 |
| 기말고사 대체 과제 |

|  |
| --- |
| 송 지민  사회학과 201721899 |

**목차**

**Ⅰ 겜블링 프로그램**

1. 프로그램 설명
2. 코드
3. 실행화면

**Ⅱ 음악파일 관리 프로그램**

1. 프로그램 설명
2. 코드
3. 실행화면

**Ⅲ 보고서를 마치며**

**Ⅰ 겜블링 프로그램**

1. 프로그램 설명

본 프로그램은 Dev C++을 통해 작성되었다.

겜블링 프로그램은 숫자 세 개를 무작위로 받고, 그 세 개의 숫자가 모두 일치하도록 나온 사람이 이기는 게임이다. 따라서 플레이어 별로 숫자를 무작위로 3개씩 받고 이를 비교하는 작업이 필요하며 3개가 일치하는 플레이어가 승자가 되도록 프로그램을 구성한다.

실제 게임과 같은 긴장감을 유지하기 위해 게임을 시작하기에 앞서 안내문을 출력하도록 하였으며 안내문을 읽고 넘어가도록 장치를 구성하였다. 또한 실제처럼 플레이어가 게임에 집중하도록 하기 위해 엔터를 재입력하여 숫자를 다시 받는 것 같은 기능은 구현하지 않았다.

첫 번째 참여자부터 시작하여 해당하는 한 바퀴를 ‘턴’으로 하였으며 각 턴이 시작할 때 턴의 수를 표시하고 게임이 끝났을 때 진행되던 턴 수를 표시하였다.

승패가 가려진 후에는 다시 게임을 시작하거나 게임을 끝낼 수 있게 하였다.

1. 코드

// 겜블링 프로그램

#include <stdio.h> // 표준입출력 헤더파일 포함

#include <stdlib.h> // 표준 라이브러리 헤더파일 포함

// 랜덤함수 사용 위함

#include <time.h> // 랜덤함수 시드값 주기 위한 타임 헤더파일

#include <math.h> // ceiling 함수 사용하기 위해 수식 헤더파일 포함

// 전역변수 선언

int i, j; // 반복제어(인덱스)변수

double turn; // 게임 턴 수를 저장할 변수

struct game { // 구조체 선언

// 구조체 멤버

int num; // 게임에 참여하는 사람 수

char name[100][100]; // 게임에 참여하는 사람의 이름

}; /\* C언어의 구조체는 객체지향언어의 클래스와 유사한 개념이다.

구조체를 이용하면 여러 종류의 데이터를 하나로 묶어서 정의할 수 있으며 struct를 통해 선언하고 구조체와 구조체를 구별하기 위한 태그를 붙인다.

중괄호 안에 포함된 변수들이 구조체의 멤버이며 구조체 멤버에는 자료형의 제약이 없으나 유일한 이름을 지녀야 한다.

구조체 선언이 끝나면 반드시 세미콜론을 붙여야 하며 구조체 선언은 변수 선언과 다르기 때문에 추가로 구조체 변수를 생성해야 한다.\*/

// 함수 원형 정의

void print\_notice(); // 게임 안내문 출력 함수

int get\_number(struct game player); // 게임 참여 인원수 입력받는 함수

void get\_name(struct game\* player); // 게임 참여 인원 이름 입력받는 함수

void print\_rand\_num(struct game\* player); // 랜덤 넘버를 인원별로 출력하는 함수

int turn\_num(double x, struct game player); // 게임 턴 수를 계산하는 함수

int restart(); // 게임 재개 의사를 묻는 함수

/\* printf()함수 혹은 scanf()함수처럼 이미 라이브러리가 존재하는 함수를 사용하기도 하지만 사용자가 직접 함수를 만들어 사용하기도 한다.

함수를 만들어 사용할 경우 코드 반복을 줄일 수 있으며 모듈화가 가능하기 때문에 유지, 보수에 있어서 이점이 있다.

main함수 또한 함수의 일종이며 코드 파악을 용이하게 하기 위해 함수 중 가장 앞에 배치한다.

그런데, 컴파일러는 메인함수를 우선실행하기 때문에 메인함수 내에 다른 함수가 존재할 경우 오류가 발생하므로 메인함수 내에서 사용되는 다른 함수들은 미리 원형을 정의해야 한다.

위의 코드는 함수 원형을 정의한 것이다. 함수 원형은 함수를 선언할 때와 동일하게 반환형, 함수의 이름, 매개변수로 이루어진다.\*/

int main(void) // 반환값이 정수형이며 매개변수가 없는 메인함수의 선언부

{

struct game player; // 구조체 변수 선언

/\* 구조체 정의와 구조체 변수 선언은 다르기 때문에 구조체를 정의했더라도 구조체 변수 선언을 반드시 해야 한다.

포인터 또한 구조체 변수로 선언할 수 있으며, 구조체 변수의 크기는 구조체 멤버의 크기를 합친 만큼이다.

소스 파일 상위에서 구조체를 정의했기 때문에 구조체 정의를 전체 소스 파일에서 사용할 수 있다.\*/

int again = 0; // 게임 재개 의사를 저장할 변수 선언 및 초기화

print\_notice(); // 게임 시작에 앞서 게임 안내 함수 호출

while (1) // 반복문을 통해 게임을 반복, 1은 TRUE를 의미하며 항상 반복함을 의미

{

player.num = get\_number(player); // 참여자 숫자를 입력받는 함수 호출, 반환값 저장

/\* 구조체 멤버인 num에 get\_number(player)의 반환값을 저장하는데, 인수는 구조체 변수의 이름을 주며 구조체 멤버를 참조할 때는 . 연산자를 쓴다.\*/

printf("\n"); // 줄바꿈

get\_name(&player); // 참여자 이름을 입력받는 함수 호출, 반환값 없음

/\* 위에서 매개변수로 구조체 변수의 이름을 준 것과 다르게 여기서는 구조체 원본을 바꾸기 위해 주소를 인수로 주었다.

이 때 구조체의 주소를 주지 않으면 함수 내에서만 이름을 받고 사라지는 오류가 발생(실행화면 참조)하므로 주의해야 한다.

즉, 위에서는 구조체 변수의 복사본을 전달했지만 get\_name(&player)에서는 구조체 변수의 원본을 전달한 것이다.\*/

printf("\n"); // 줄바꿈

print\_rand\_num(&player); // 랜덤넘버를 참여자별로 출력하여 게임을 진행하는 함수 호출

/\* 여기서도 마찬가지로 인수로 주소를 주어 구조체 멤버의 원본을 바꾸도록 하였다. 반환값이 없으므로 인수를 주어 호출만 한다.\*/

printf("\n게임은 %d 턴 진행되었습니다.\n\n", turn\_num(turn, player)); // 승부가 났을 때 게임 턴 수를 알림

/\* 턴 수를 알리기 위해 턴 수를 계산하는 함수를 호출한다. 이 함수에 필요한 매개변수는 총 플레이 수 및 턴수를 저장할 변수, 플레이어 수이다.

반환값이 있기 때문에 계산 후 반환값을 받아 출력하면 되므로 원본을 전달할 필요가 없다.\*/

again = restart(); // 사용자로부터 게임 재개 의사를 받는 함수를 호출

/\* 위 코드의 함수는 매개변수가 없으므로 인수를 주지 않았으며 반환값을 again에 저장하여 사용한다.\*/

if (again == 1) // 게임 재개 의사가 있을 때

continue; // 다음 반복을 시작하기 위해 continue문 사용

else // 게임 재개 의사가 없을 때

break; // 항상 참인 while문을 빠져나가 종료

}

return 0; // 0을 반환하여 메인함수 끝내기

}

void print\_notice() // 반환값과 매개변수가 없는 게임 안내 함수

{

char next = { '\0' }; // 사용자가 안내문을 읽고 게임을 진행할 수 있도록 하기 위한 변수 선언 및 초기화

printf("☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆\n"); // 겜블링 안내문 출력 시작

printf("★ 겜 블 링 안 내 ★\n");

printf("☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆\n"); // 겜블링 안내 출력

printf("\n※원활한 플레이를 위하여 안내사항을\n '반드시' 숙지해주시길 바랍니다.\n"); // 주의문 출력

printf("\n※세 개의 숫자가 모두 일치하는 사람이 이깁니다.\n"); // 게임 룰 안내

printf("\n※선수는 최소 한 명, 최대 백 명입니다.\n"); // 참여 가능한 인원 수 안내

printf("\n※자신의 턴에서 <Enter>키 외의 키를 입력하시면 \n 그 턴은 '기권으로 간주'하니 주의 바랍니다.\n"); // 게임 진행 시 주의사항 출력

/\* 게임을 시작하기에 앞서 안내문을 출력하게 하였으며 긴장감 조성을 위해 경고문을 삽입하였다.\*/

printf("\n※넘어가시려면 아무 키나 입력해주세요."); // 다음으로 넘어가기 위한 안내 출력

scanf("%c", &next); // 사용자로부터 키를 입력받아 넘어가기

/\* 사용자가 안내문을 읽지 않았을 경우 원활한 게임이 어려울 수 있기 때문에 안내문을 읽도록 장치를 구성하였다.\*/

if (next != '\0') // 사용자로부터 키를 입력받았을 때(널 포인터 외의 키 입력 시, 즉 키 입력 시)

{

printf("\n ♬즐거운 게임 되시길 바랍니다.♪\n"); // 문구를 출력

printf("----------------------------------------------\n\n"); // 안내문과 게임 화면을 나누기

}

}

int get\_number(struct game player) // 반환값이 정수이고 매개변수로 구조체 변수를 갖는 함수, 사용자로부터 플레이어 수 입력 받음

{

printf("겜블링 게임에 참여할 선수 숫자>>"); // 플레이어 수 입력 안내

scanf("%d", &player.num); // 플레이어 수 입력받아서 구조체 멤버 num에 저장

while (player.num < 1 || player.num > 100) // 플레이어가 참가 불가능 인원수를 입력했을 때(0명이하, 100명 초과)

{

printf("\n선수 숫자는 1명부터 100명까지 가능합니다.\n"); // 안내 출력

printf("다시 입력해주세요."); // 재입력 안내

scanf("%d", &player.num); // 다시 입력받기

getchar(); // 참여자가 문자를 입력했을 경우 버퍼를 비우기

} // 사용자가 0부터 100사이의 숫자를 입력하지 않을 때 계속 반복되게 함

return player.num; // 사용자로부터 받은 수를 반환

}

void get\_name(struct game\* player) // 반환값이 없으며 매개변수로 포인터를 가지는 함수, 플레이어 이름 입력 받음

{

for (i = 0; i < player->num; i++) // 반복문을 사용해 플레이어 수만큼 반복

// 위에서 포인터를 사용했기 때문에 구조체 멤버를 가리킬 때도 -> 를 사용

{

printf("%d번째 선수 이름>>", i + 1); // 선수 이름 입력 안내, i는 0부터 시작하므로 i+1

scanf("%s", &player->name[i][0]); // 선수 이름 입력받기

/\* 구조체 멤버 name은 2차원 배열로 선언하였기 때문에 for문의 인덱스 변수를 이용하여 구조체 멤버 num의 수만큼 플레이어의 이름을 저장할 수 있다.

이 때 사이즈가 100이므로 100명 입력이 가능하며 각 플레이어의 이름은 최대 100자(한글은 상이할 수 있음)이다.\*/

}

}

void print\_rand\_num(struct game\* player) // 플레이어 별로 랜덤넘버를 3개 생성하고 게임 반복하는 함수

{

int k = 0; // do while문 반복을 제어하기 위한 변수 선언 및 초기화

int number[3] = { 0 }; // 랜덤넘버를 저장하기 위한 배열 선언 및 초기화

char enter = '\0'; // 자신의 턴에서 랜덤넘버를 받기 위한 변수 선언 및 초기화

/\* 선언 시에 초기화하지 않으면 쓰레기값이 생성될 수 있으므로 초기화 하는 습관을 들이는 것이 좋다.\*/

turn = 0; // 전역변수로 선언된 변수에 0 입력

srand(time(NULL)); // 랜덤함수의 난수를 더 다양하게 받기 위하여 시드값을 time(NULL)로 지정

/\* srand()함수 미사용 시 항상 4번째 차례에서 333이 나와 게임이 끝나는 것을 확인하였다. 따라서 실제로 게임을 진행하기 위해서는 srand() 함수를 사용해야 한다.

srand()함수는 stdlib.h에 포함되어 있다.\*/

do // do while 문은 조건에 상관 없이 우선 중괄호 안의 내용을 한 번 실행한다는 점에서 while문과 다름

{

printf("<%d턴 입니다.>\n\n", turn\_num(turn, \*player) + 1); // 한 바퀴 돌 때마다 턴 수를 출력

/\* 턴 수 출력을 위해 턴 수를 계산하는 함수를 호출하는데, 이 때 턴 수가 0으로 초기화되어 있기 때문에 반환값에 0을 입력해야 한다.

또한 print\_rand\_num(struct game\* player)에서 포인터 형식으로 매개변수를 설정했기 때문에 turn\_num의 인수 또한 포인터 형식으로 해야 한다.\*/

for (i = 0; i < player->num; i++) // 플레이어 수만큼 반복하기

/\* 매개변수가 포인터 형이기 때문에 마찬가지로 -> 를 이용하여 구조체의 멤버를 가리켜 준다.\*/

{

turn++; // 턴이 시작할 때 총 턴의 수를 1씩 증가시켜 준다.

fflush(stdin); // 버퍼비우기

/\* 버퍼를 비우지 않았을 때 엔터를 입력하지 않았을 때도 랜덤넘버를 출력하는 등의 문제가 발생하여 버퍼를 비웠다.

이는 특히 문자, 문자열 입력 시 자주 발생하는 오류이기 때문에 문자열 입력 시 주의해야 한다.

다만, fflush(stdin)은 표준이 아니기 때문에 오류가 발생할 수 있음을 유의한다.

fflush(stdin) 대신 while(getchar() != '\n')을 사용하기도 하는 것으로 보인다.\*/

printf("[%s]:<Enter>", player->name[i]); // 참조 연산자 이용해 플레이어 이름을 출력

/\* -> 연산자는 구조체 포인터로 구조체 멤버를 참조할 때 사용한다. 즉, player가 가리키는 곳의 name라는 의미이다.\*/

scanf("%c", &enter); // 엔터 입력받기

if (enter == '\n') // 만약 사용자가 엔터를 입력했을 때(엔터는 개행문자와 같다)

{

printf("\n "); // 출력형태를 만들기 위해 개행, 탭

for (j = 0; j < 3; j++) // 0부터 2까지 3번 반복

{

number[j] = rand() % 3 + 1; // 차례대로 랜덤넘버 3개를 받아 저장

printf(" %d", number[j]); // 랜덤넘버를 차례대로 출력

}

if (number[0] == number[1] && number[1] == number[2]) // 만약 세 랜덤넘버가 같다면

/\* 이 때 number[0] == number[1] == number[3]과 같은 형태로 비교할 수 없음을 유의한다.

이는 ==이 삼항 연산자가 아니기 때문으로 보인다.\*/

{

printf(" %s 님이 이겼습니다!\n", player->name[i]); // 세 개의 수가 모두 일치할 때 그 플레이어의 이름을 출력

k = 1; // k 값을 1로 바꿈

break; // for 문을 빠져나감

}

else // 세 랜덤넘버가 갖지 않을 때

printf(" 아쉽군요!\n"); // 이기지 못했음을 나타내는 문구 출력

}

else // 플레이어가 엔터 외의 키를 입력했을 때

{

printf("\n ※기권하셨습니다.※\n\n"); // 게임 안내문의 내용대로 기권으로 처리

continue; // 다음 반복을 시작

}

}

} while (k != 1); // k=1일 때는 세개의 수가 모두 일치할 때이므로 그렇지 않을 때는 계속 반복

}

int turn\_num(double x, struct game player) // 턴 수를 계산하는 함수, 실수형 변수와 구조체 변수를 매개변수로 가짐

{

x = x / player.num; // 턴 수를 구하기 위해 플레이어 수로 총 반복된 횟수를 나눔

/\* 턴 수는 플레이어가 한 번씩 자신의 차례를 지난 것을 의미하므로 플레이어 수로 나눠준다.\*/

return ceil(x); // x값을 ceil()함수로 하여 반환

/\* ceil()함수는 반올림 함수와 유사하나 소수점이 있는 경우 값의 크기에 상관 없이 값을 올려 정수로 만든다.

이 함수는 math.h에 정의되어 있기 때문에 전처리문에서 포함하여 선언하였다.\*/

}

int restart() // 반환값이 정수형이며 매개변수가 없는 게임 재개 의사 함수

{

int y = 0; // 게임 재개 의사를 묻고 저장할 변수 선언 및 초기화

printf("※게임 재개를 원하십니까?※\n"); // 게임 재개 의사 묻기

printf("게임 재개는 1번, 마치시려면 1번 이외의 키를 입력해주세요."); // 재개 방법 안내하기

scanf("%d", &y); // 재개 의사 입력받아 y에 저장하기

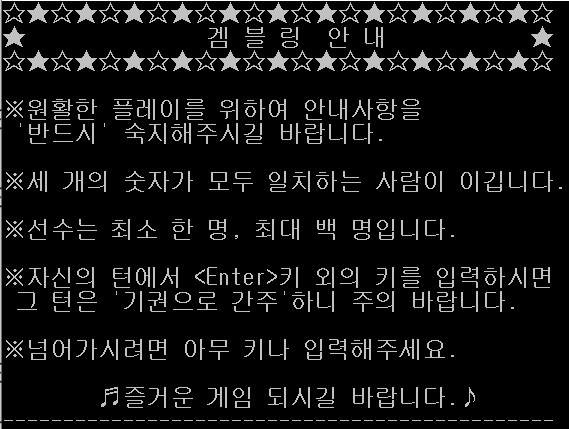
getchar(); // 사용자가 문자를 입력했을 때 오류를 막기 위해 버퍼 비우기

printf("\n"); // 줄바꿈

return y; // y반환하기

}

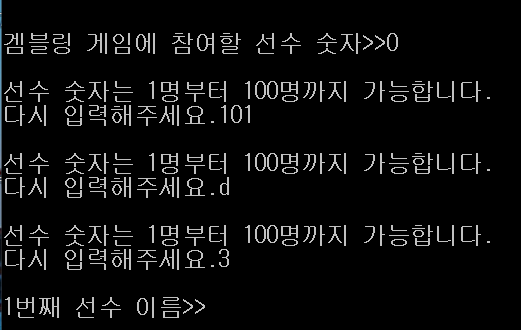
1. 실행화면



왼쪽의 실행화면은 게임 시작 전 안내문을 출력한 화면이다.

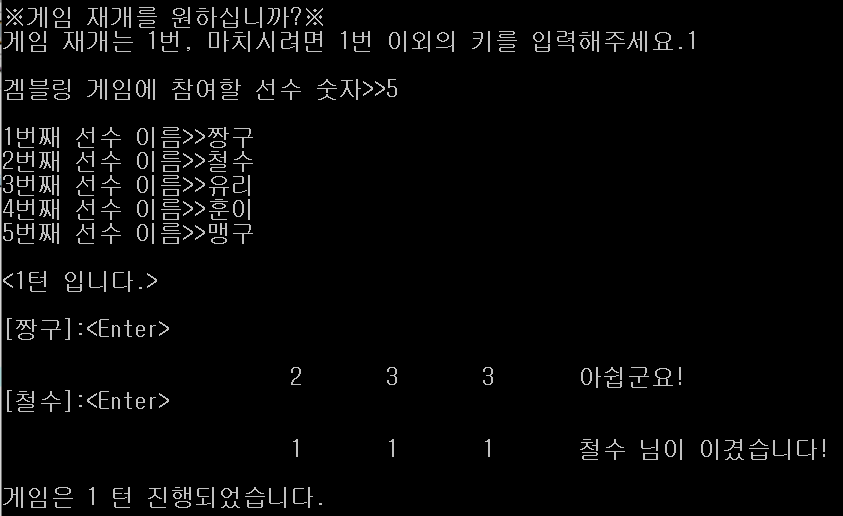
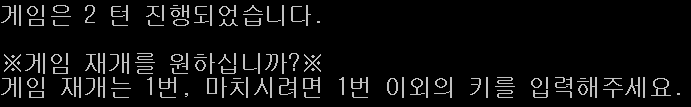
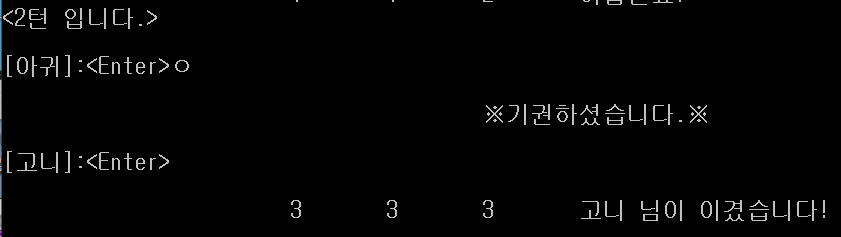
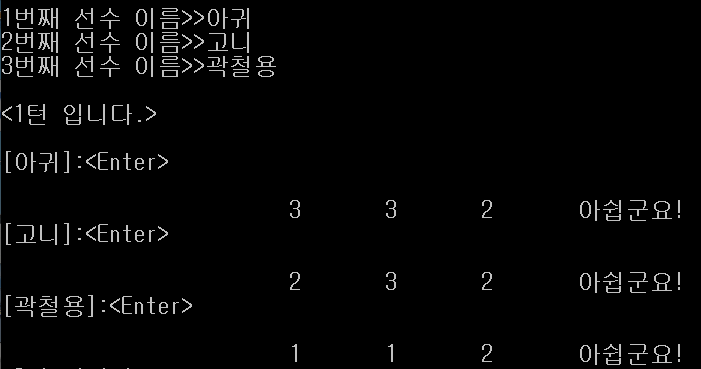
겜블링의 간단한 내용을 설명하고 참여 인원수와 플레이 방법, 주의사항을 설명하였다.

이 때 안내문 출력 후 바로 게임으로 넘어가지 않고 플레이어가 키를 입력하게 하였다.



왼쪽의 화면은 플레이어 인원 수 입력 화면이다.

참여 가능한 플레이어 수는 1부터 100사이이기 때문에 그 외의 것을 입력할 경우 오류 처리를 하여 다시 입력받도록 하였다.



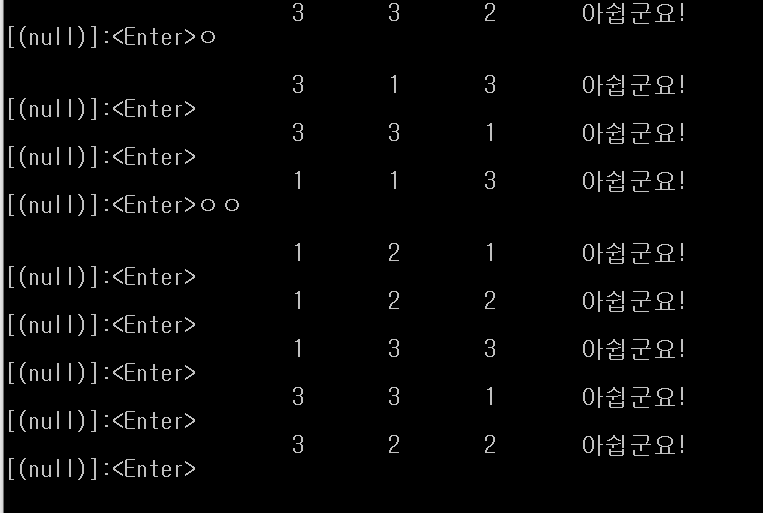
왼쪽의 화면은 1부터 100 사이의 인원수 입력 후 나타나는 화면이다. 이름을 차례대로 입력하고 게임을 시작하며, 세 명 모두 세 숫자가 일치하지 않아 ‘아쉽군요!’가 출력된다.

위의 화면은 2턴째가 되고 아귀가 엔터 외의 키를 입력해 기권하였을 때의 화면이다.

아귀의 기권 후 고니의 턴에서 고니의 수가 일치하여 ‘고니 님이 이겼습니다!’가 출력되었다.

고니의 승리로 게임이 끝난 후, 진행된 턴 수를 출력한다. 또한 사용자에게 게임 재개 의사를 묻고 이를 입력 받는다.

게임 재개를 입력 받았을 때는 다시 참여자 수와 이름을 받고 게임을 처음부터 진행한다.



위의 화면은 get\_name()함수에서 포인터를 사용하지 않았을 때 구조체 변수의 복사본을 전달받아 플레이어 이름이 저장되지 않은 오류 화면이다.

또한 버퍼를 비우지 않아 엔터 이외의 키를 눌러도 차례가 반복되는 오류를 포함하고 있다.

위의 화면은 srand()함수를 사용하지 않았을 때 4번째 차례에 게임이 끝나는 화면이다.

**Ⅱ 음악파일 관리 프로그램**

1. 프로그램 설명

프로그램은 Visual Studio(2019)로 작성하였다. 간단한 음악 파일 관리 프로그램으로, 음악의 제목, 가수, 위치, 장르를 입력받아 저장하고 출력, 검색 등을 수행하는 프로그램이다.

구조체 배열을 사용하여 하나의 구조체에 여러 개의 음악을 저장할 수 있게 하였으며 사용자가 입력하는 숫자에 따라 다른 처리가 이루어지도록 switch문을 이용하였다. 기본기능 외 삭제 기능을 추가하였으며 사용자가 메뉴(1부터 5까지) 외의 키를 입력했을 때 오류를 안내하고 메뉴를 다시 입력받는 기능을 구현하였다.

또한 프로그램은 사용자가 종료를 입력하기 전까지 계속적으로 반복되며 입력한 음악 파일은 누적하여 저장되고 출력된다.

1. 코드
2. 실행화면

**Ⅲ 보고서를 마치며**

시간 관계상 구현하지 못한 기능도 다수 존재하고 스스로 생각하기에 기능의 완성도가 아쉬운 측면이 있는 것이 사실이다. 특히 enum, typedef 의 경우 아직 학습이 이루어지지 않아 그 장벽이 높게 느껴졌다. 머릿속에 알고리즘을 구현하고 그것을 실제로 옮기는 능력이 아직은 부족하기에 이번 학기가 끝나고 방학 동안 꾸준히 연습해야겠다는 생각이 들었다.

앞으로 본 프로그램을 수정할 때 추가, 보완하고 싶은 기능은 겜블링의 경우 더욱 사실적으로 구현하기 위해 인터넷 카드 게임처럼 포인트를 걸고 지면 포인트가 삭감되고 이기면 포인트를 얻는 것과 같은 기능이다. 음악 관리 프로그램의 경우, 구현할 수 있는 기능은 더욱 다양하기에 후에 계속해서 수정하면서 실제 음악을 불러오고 파일을 수정할 수 있도록 고쳐나가고 싶다.

프로그램을 구현하고 디버깅하며 많은 시행착오를 겪었고, 답답함도 컸지만 디버깅에 성공하면서 프로그램이 제대로 작동될 때 기쁨을 느끼게 되었다. 또한 처음에는 문자를 입출력하고 정수와 실수를 계산하는 것도 헷갈려 했는데, 포인터와 구조체를 통해 기능을 구현할 정도로 발전할 수 있었던 것에 감사한 마음이 들었다. 한 학기 동안 프로그래밍 언어를 수강하면서 C언어를 처음으로 배우게 되어, 생소한 분야이다보니 어려움이 컸지만 보람 또한 크게 느껴지는 시간이었다고 생각한다. 앞으로도 포기하지 않고 정진하여 많은 사람들에게 도움을 주는 개발자가 되고 싶다.