CSED101. Programming & Problem solving

Fall, 2020

Programming Assignment #3

(65 points)

박채용 (pcy8201@postech.ac.kr)

- **제출 마감일: 2020.12.02 23:59**
- **개발 환경: Windows Visual Studio 2019**

- **제출물**
  - **C Code files (assn3.c)**
    - 프로그램의 소스 코드를 이해하기 쉽도록 반드시 **주석**을 붙일 것.
  - **보고서 파일 (assn3.docx, assn3.hwp 또는 assn3.pdf)**
    - AssnReadMe.pdf 를 참조하여 작성할 것.
    - 명예서약(Honor code): 표지에 다음의 내용을 포함한다. "나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다." 보고서 표지에 명예서약이 없는 경우는 과제를 제출하지 않은 것으로 처리한다.
    - 소스코드와 보고서 파일을 LMS를 이용하여 제출한다.

- **주의사항**
  - 각 문제에 해당하는 요구사항을 반드시 지킬 것.

  - 모든 문제의 출력 형식은 아래의 예시들과 동일해야 하며, 같지 않을 시는 감점이 된다.
  - 각 문제에 제시되어 있는 파일이름으로 제출 할 것. 그 외의 다른 이름으로 제출하면 감점 또는 0점 처리된다.
  - 컴파일 & 실행이 안되면 무조건 0점 처리된다
  - 추가 기능 구현에 대한 추가 점수는 없다.
  - 과제를 구현하는데 있어 필요한 개념들은 마지막 섹션인 힌트에서 자세하게 다룬다.

  - **하루 late시 20%가 감점되며, 3일 이상 지나면 받지 않는다. (0점 처리)**
  - 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 'POSTECH 전자컴퓨터공학부 부정행위 정의'를 따른다. (LMS의 과목 공지사항의 제목 [document about cheating]의 첨부파일인 disciplinary.pdf를 참조할 것.)

Problem: 길건너 친구들

[문제]

길건너 친구들은 스마트폰으로 할 수 있는 게임으로, 캐릭터가 여러 장애물과 부딪히지 않고 무사히 앞으로 전진하는 게임이다. 본 과제에서는 이 게임의 큰 틀을 가져와 간소화 된 버전의 게임을 만들어, 캐릭터를 마지막까지 단계까지 무사히 전진시킬 수 있도록 한다.

그림 1. 길건너 친구들

- 길건너 친구들은 구글 플레이스토어에서 받아서 해볼 수 있다.
  - 게임의 진행 영상은 다음의 링크에서 볼 수 있다.
  - https://youtu.be/FdgxK4gR3eg

[목적]

본 과제는 다음의 내용을 목적으로 한다.

- 파일 입출력의 사용법을 익힌다.
- 포인터와 동적 할당 사용법을 익힌다.
- 간단한 게임을 구현함으로 알고리즘 구현 능력을 기른다.

[주의사항]

- 파일 이름은 "assn3.c"로 저장 한다.
- 보고서 이름은 "assn3.docx", "assn3.hwp" 또는 "assn3.pdf"로 저장한다.
- 전역 변수, goto 문, 구조체, 객체지향프로그래밍의 개념은 사용할 수 없다.
- 기능 구현 시 실시간 사용자 입력을 받도록 한다. (힌트를 참고할 것)
- 과제는 2차원 배열을 동적으로 할당하고 사용하기 위한 과제이므로, 맵은 반드시 2차원 배열을 동적으로 할당 한 뒤 사용 및 해제 한다. (힌트를 참고할 것)
  - 동적 할당을 이용하지 않고 크기가 정해진 2차원 배열을 선언 후 사용 할 시 0점 처리한다.
- 모든 기능을 main 함수에서 구현한 경우 감점 처리한다.
  - 기능적으로 독립됐거나 반복적으로 사용되는 기능은 사용자 함수를 정의해서 구현한다.
- 프로그램 구현 시, main()함수를 호출하여 사용하지 않는다.
- 문제의 출력 형식은 채점을 위해 아래의 실행 예시와 일맥상통하게 출력한다.
  - 비슷하거나 형식이 이해된다면 감점하지 않는다.

[게임 규칙 설명]

길건너 친구들은 맵에서 캐릭터가 다양한 물체들을 피해가면서 앞으로 나아가는 게임이다. 게임 환경은 아래의 그림과 같으며, 정해진 맵의 끝까지 도달하게 되면 게임은 종료된다. (여기서 "정해진 맵"은 텍스트 파일로 부터 읽고/생성한 전체의 맵을 의미한다.)

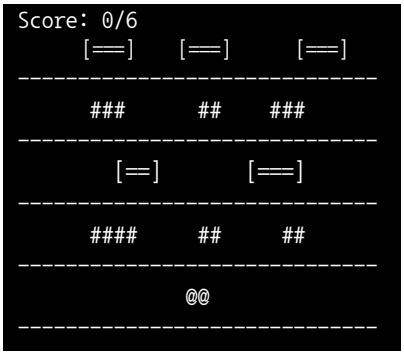


그림 2. 본 과제의 출력 예시

- 게임은 가로 5 x 세로 30 의 고정된 크기의 맵을 가진다.
- 캐릭터는 초기에 가로 14 지점, 세로 0 지점에서 시작한다.
- 좌측 상단에는 획득한 점수 (앞으로 나아간 횟수)를 나타낸다.
- @@: 캐릭터를 나타내며, w키를 이용하여 전진, a/d키를 이용하여 좌/우로 움직일 수 있다.
- #: 나무를 나타내며, 움직이지 않는 오브젝트이다. 하지만 캐릭터는 나무를 통과하지 못한다.
- [==]: 자동차를 나타내며, 매 프레임 오른쪽으로 움직이는 오브젝트이다. 캐릭터가 자동차와 충돌하면 게임에서 패배하게 되고 게임은 종료된다.
- 본 과제의 이해를 돕기 위한 대한 영상은 다음의 링크에서 볼 수 있다.
  - 난이도 Normal 선택 후 게임 승리, Hard 선택 후 게임 패배를 보여준다.
  - https://drive.google.com/file/d/1ReyQZEY1G2SGPE3NFuBevluHVxczDt9i/view?usp=sharing
  - 영상 재생 실패 시 다운받아서 플레이 하시면 됩니다.

[과제 설명]

프로그램의 진행 방식은 다음의 흐름도와 같으며, 크게 총 4가지 단계로 이루어져 있다. 각 단계에 대한 설명은 순서대로 다음 장부터 설명된다.

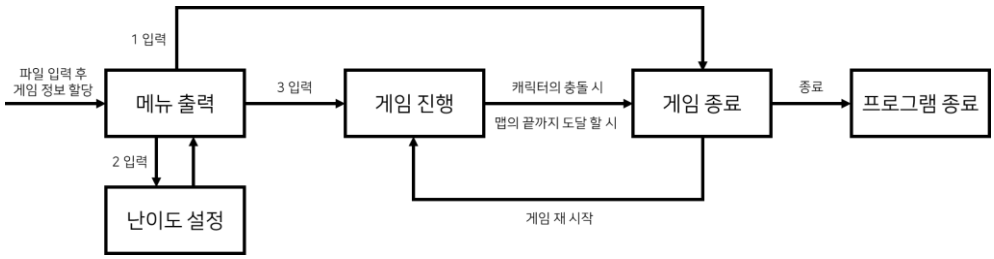


그림 3. 본 과제의 흐름도

0. 맵 정보 생성
1. 메뉴 출력
2. 난이도 설정
3. 게임 진행
4. 게임 종료

[0. 맵 정보 생성]

■ 파일 입력

게임에 이용 될 맵의 정보는 txt파일을 이용하여 입력 받는다. 프로그램 실행 시, map.txt 파일에서 읽어오도록 프로그램을 작성한다. 단, 파일이 없는 경우 "File open error!!"라는 메시지 출력 후, 프로그램을 종료한다. 입력 받을 txt파일은 다음의 양식을 따른다.

map.txt
6 0 0 1 3 2 2 1 3 2 3 1 3

- 첫째 줄에는 항상 전체 단계 수 (N) 를 나타낸다.
- 둘째 줄에는 항상 0 0을 입력한다.
  - 캐릭터의 시작 지점은 가로 14, 세로 0 지점으로 지정되어 있으며, 첫번째 줄에는 아무 오브젝트도 존재하지 않는다.
- 다음 줄부터는 오브젝트에 대한 정보들이 N-1개 주어진다.
- A B라고 주어 질 때, A는 오브젝트의 종류를 (1-나무, 2-자동차), B는 오브젝트의 개수를 나타낸다.
  - 예를 들어, 세번째 줄의 1 3의 경우, 그 단계에서 총 3개의 나무가 한 줄에 등장한다.
  - 오른쪽 상단의 예시로 구성된 화면은 그림7에 해당한다.
- 본 문서는 함께 제공 된 map.txt 파일을 통해 설명되었으며, 함께 제공 된 예시 동영상은 map_long.txt를 이용하여 촬영되었다.

[1. 메뉴 출력]

본 단계에서는 게임 시작을 위한 메뉴를 출력한다. 메뉴 출력 화면은 다음과 같다.

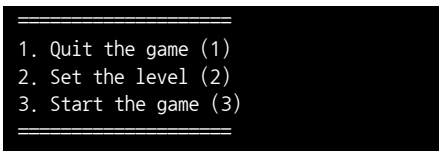


그림 4. 메뉴 화면

- 1을 입력할 시 프로그램을 종료한다. (입력에 대한 방법은 힌트를 참고한다.)
- 2를 입력할 시 난이도를 조절한다.
  - 난이도를 조절 한 후 다시 메뉴화면으로 돌아올 수 있도록 반복문(while)을 이용하여 메뉴를 출력한다.
- 3을 입력할 시 게임을 시작한다.
  - 난이도 설정을 하지 않을 경우 기본적으로 Normal로 게임이 진행된다.
  - 게임이 끝난 뒤 다시 메뉴화면으로 돌아와야 하므로 반복문을 이용하여 게임을 진행한다.
- 1, 2, 3외의 입력에 대해서는 고려하지 않는다.
- 사용자입력은 화면에 출력 할 필요 없다.

- 반복문을 사용하는 구조도는 다음의 그림과 같이 나타낼 수 있다.
  - 이해를 돕기 위한 예시 일 뿐, 다음과 같은 구조를 가지지 않아도 상관 없다.

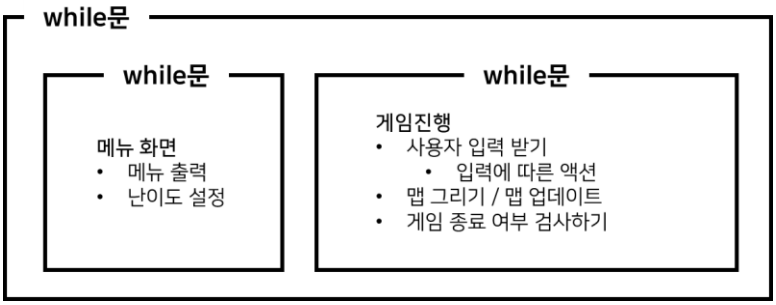


그림 5. 본 과제의 큰 구조도 예시

[2. 난이도 설정]

본 단계에서는 난이도 조절을 위한 메뉴를 출력한다. 메뉴 출력 화면은 다음과 같으며, 난이도는 총 세가지를 선택할 수 있다. 원하는 난이도에 해당하는 번호를 입력 받으면, 난이도에 해당하는 변수 설정을 진행하고 메뉴화면으로 넘어간다.

그림 6. 난이도 설정 화면

■ 난이도 조절 방법

난이도는 게임의 화면 업데이트 속도를 변경하여 조절한다. 즉 Sleep 함수의 대기 시간을 조절함으로 화면의 업데이트 속도를 조절 할 수 있다. (Sleep 함수 및 화면 업데이트는 힌트 참고)

- Easy를 선택할 시 (1번 입력 시)
  - Sleep 시간을 200으로 설정한다.
- Normal을 선택할 시 (2번 입력 시)
  - Sleep 시간을 100으로 설정한다.
- Hard를 선택할 시 (3번 입력 시)
  - Sleep 시간을 50으로 설정한다.
- 키보드 입력에 대한 출력은 하지 않아도 된다.
- 1, 2, 3외의 입력에 대해서는 고려하지 않는다.

[3. 게임 진행]

본 단계는 게임을 진행하는 단계이다. 기본적으로 맵을 출력하고, 매 프레임 마다 자동차([==])는 한 칸씩 오른으로 이동한다. 기존 과제와는 다르게 콘솔창에 나오는 화면을 지우고 다시 그리는 방식을 이용한다 (이는 힌트를 참고한다.). 사용자는 키보드 입력을 통하여 캐릭터 (@@)를 움직이며 앞으로 나아갈 수 있다. 이동 할 때, 나무(#)로는 이동 할 수 없으며, 자동차에 부딪히지 않고 최종단계까지 나아갈 경우 게임에서 승리한다. 움직이다가 자동차와 부딪히거나, 가만히 있다가 자동차에 부딪히는 경우 게임에서 패배한다. 맵을 생성하는 방식에 대해서는 2차원 배열의 동적 할당을 이용하며 힌트를 참고한다.

- 게임 진행 방식은 위에 있는 동영상을 참조한다.

■ **화면 출력 양식**

화면 출력은 다음과 같은 양식으로 출력한다. 제일 윗줄에 현재 캐릭터가 나아간 단계와 전체 단계를 다음과 같은 양식으로 출력해주고, 그 아래 5줄의 맵을 출력한다. 맵 관리에 대해서는 힌트를 참고한다.

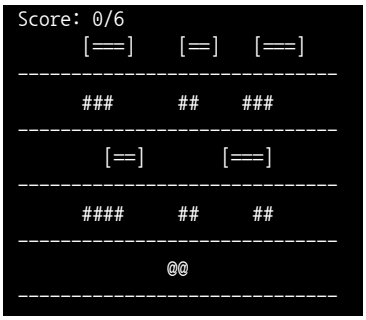


그림 7. 화면 출력 양식

■ **오브젝트 설명**

게임에서는 나무, 자동차로 이루어 진 2가지 종류의 오브젝트가 존재한다. 오브젝트들은 맵의 한 줄이 생성 될 때 랜덤한 위치와 랜덤한 크기로 생성되도록 한다. 오브젝트의 생성 방법은 자유롭게 하며 오브젝트에 대한 설명과 그 성질은 다음과 같이 설정한다.

- 나무 (#)
  - 나무는 시간이 지나도 (프레임에 상관없이) 항상 고정 되어있는 오브젝트이다.
  - 나무의 크기는 2-5사이의 랜덤한 크기를 갖는다.
- 자동차 ([==])
  - 자동차는 매 프레임마다 한 칸씩 오른쪽으로 이동하는 오브젝트이다.
  - 자동차가 오른쪽 끝에 도달하면 다시 왼쪽 끝에서 시작되도록 한다.
    - ◆ 그림 11의 가운데 화면 예시를 참고한다.
  - 자동차의 크기는 4-7 사이의 랜덤한 크기를 갖는다.
    - ◆ 크기 4: [==], 크기 5: [===], 크기 6: [====], 크기 7: [=====]
- 오브젝트와 오브젝트의 사이 거리가 너무 좁으면 캐릭터가 이동하기 힘들기 때문에, 오브젝트 사이의 거리는 적어도 4-6칸을 띄우도록 한다.
  - 띄우는 거리에 대해서는 정해진 규칙이 없다.

■ **캐릭터의 이동**

캐릭터는 키보드의 a/d키를 이용하여 왼쪽/오른쪽으로 이동 할 수 있고, w키를 이용하여 앞으로 전진할 수 있다 (뒤로 가는 기능은 넣지 않는다.). 사용자 입력 값을 받으면 움직임에 대해서 충돌을 검사한 후 충돌이 발생하지 않으면 캐릭터를 움직인다. 이때, 캐릭터는 다음과 같이 상황에 따라 각기 다른 움직이는 양상을 갖는다.

- 좌/우로 이동 할 경우
  - 이동하고자 하는 방향으로 맵에서 캐릭터를 이동한다.

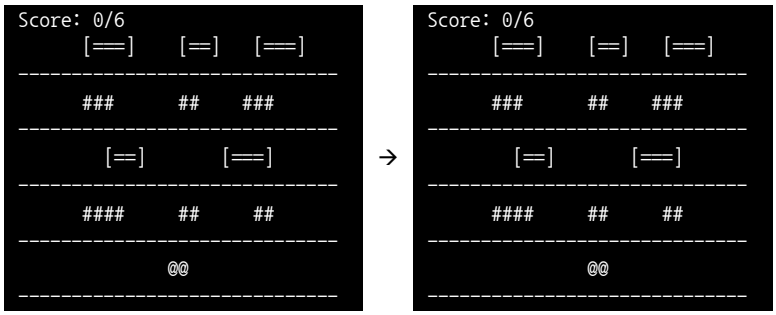


그림 8. 캐릭터를 오른쪽으로 한 칸 이동한 경우

- 앞으로 이동 할 경우
  - 캐릭터의 위치는 그대로 둔 채 전체 맵을 한 칸 밑으로 내린다. (이를 통해 캐릭터가 한 칸 앞으로 전진할 듯 한 효과를 만든다.
  - 앞으로 전진함과 동시에 점수를 1점 올린다.

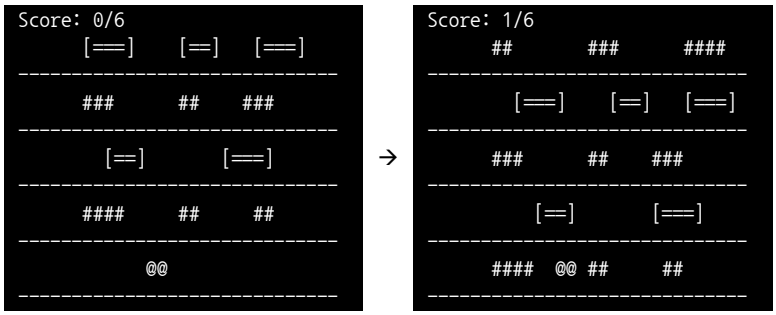


그림 9. 캐릭터를 앞으로 한 칸 이동한 경우, 맵을 아래로 이동시킨다.

■ 매 프레임마다 해야하는 작업

매 프레임마다 맵을 업데이트 해줘야 한다. 이때 업데이트 해야 할 내용은 다음과 같다.

- 나무 (#)의 경우 그대로 유지한다.
- 자동차 ([==])의 경우 매 프레임마다 오른쪽으로 한 칸 움직인다.
- 캐릭터가 앞으로 전진 할 경우, 맵을 아래로 한 칸 내린다. 이때, 가장 아래 줄은 삭제 (메모리 할당 해제)를 통해 제거한다. 또한, 나머지 줄들은 각 배열의 첫번째 주소값을 아래의 변수 공간에 복사해줌으로 한 칸씩 내려준다. 마지막으로, 가장 윗줄에는 다시 동적 할당을 통해 새로운 1차원 배열을 할당하고, 그 단계에 맞는 새로운 오브젝트들을 생성한다.
  - 꼭, 메모리 해제 및 할당 과정을 거쳐서 맵을 업데이트 한다.
  - 동적 할당을 이용하지 않고, 2 차원 배열에서의 요소들의 복사로만 진행 할 시 감점한다.
  - 다음은 맵 업데이트의 예시이다.

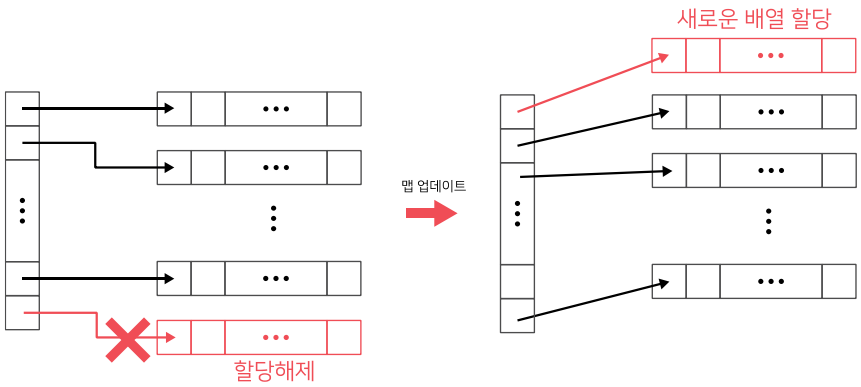


그림 10. 맵의 움직임을 위한 메모리 해제 및 재 할당 방법

■ 캐릭터의 이동 제한

캐릭터는 나무(#)를 만나면 이동 할 수 없으며, 자동차와 충돌 할 경우 게임을 종료한다. 충돌을 검사하는 함수를 이용하여 쉽게 구현할 수 있으며, 충돌 검사에 대한 설명은 힌트를 참고한다. 캐릭터가 맵 바깥으로 이동하지 못하도록 왼쪽 (혹은 오른쪽) 가장자리에 다가가면 더 이상 움직일 수 없도록 한다.

[4. 게임 종료]

본 단계에서는 게임이 종료되었을 때의 상황을 나타낸다. 게임이 종료된 후 엔터키를 입력 받아 초기 메뉴화면으로 돌아 간다. (엔터키 외의 입력에 대해서는 고려하지 않는다.)

■ 게임 종료 조건

게임은 다음과 같은 상황에서 종료된다.

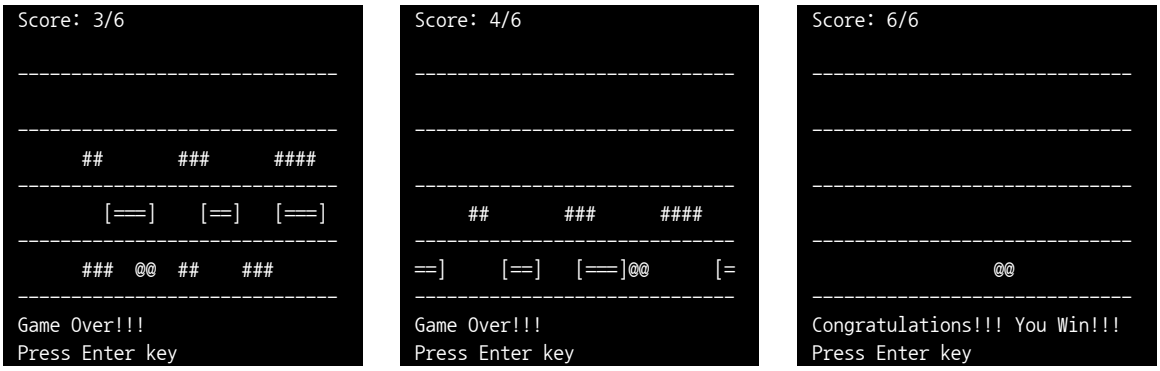


그림 11. 게임의 종료 화면. 게임의 패배 (왼쪽, 중앙), 게임의 승리 (오른쪽). 캐릭터가 움직여서 자동차와 충돌 한 경우 (왼쪽), 다가오는 자동차와 충돌 한 경우 (중앙), 마지막까지 무사히 도달 한 경우 (오른쪽)

- 캐릭터가 자동차와 충돌한 경우
  - 그림 11의 왼쪽 화면은 'w'키를 입력하여 캐릭터가 자동차와 충돌한 경우, 그림 11의 가운데 화면은 캐릭터와 다가오는 자동차가 충돌 한 경우이다.
  - 맵 업데이트를 중단하고 게임이 종료되었다는 문구를 출력한다.
  - 엔터키를 입력 받아 초기 메뉴화면으로 돌아 갈 수 있도록 한다. (scanf함수 이용)
- 입력 받은 모든 맵을 캐릭터가 다 지나갔을 경우 (그림 11의 오른쪽 화면)
  - 맵 업데이트를 중단하고 승리 문구를 출력한다.
  - 앞 경우와 마찬가지로, 엔터키를 입력 받아 초기 메뉴화면으로 돌아갈 수 있도록 한다.
- 캐릭터가 나무에 둘러 쌓여 더 이상 움직일 수 없는 경우
  - 본 상황에 대해서는 고려하지 않는다.

[문제 힌트]

■ 함수를 잘 이용하기

위에 언급된 기능들의 대부분은 각각의 단일 기능을 하는 함수로 만들면 수월하게 진행 할 수 있다. 구현하면 좋은 함수/이용하면 좋은 변수들은 힌트의 마지막에 언급하도록 한다.

■ 화면 지우기

아래의 system 함수를 이용하여 콘솔 화면에 출력된 내용을 지울 수 있다. 이를 이용하면 콘솔창에 있는 출력 화면을 지우로 재 출력하는 작업을 반복 할 수 있다.

<stdlib.h> 헤더를 추가 한 후 이용한다.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello Assn3\n");
    system("cls");
    return 0;
}
```

■ 매 프레임 화면 업데이트 방법

아래의 windows 함수를 이용하면 일정 시간 동안 작업을 대기 할 수 있다. 본 함수를 통해서 화면의 업데이트 속도를 조절 할 수 있다. 난이도 설정의 경우 이 변수에 들어가는 값을 조절한다.

<windows.h> 헤더를 추가 한 후 이용한다. 대기 시간의 단위는 ms이다.

```
while (1)
{
    pritnf("Frame\n");
    Sleep(100);
}
```

■ 랜덤 변수 사용

파일로부터 입력받은 오브젝트의 랜덤한 생성을 위해, 랜덤 함수를 이용해야한다. rand() 함수를 이용하여 난수를 생성 할 수 있다. 그러나 그냥 이용 할 경우, 항상 같은 난수를 생성하기 때문에 srand() 함수를 이용하여 매번 다른 난수를 생성해야 한다.

다음과 같은 방법으로 난수를 생성 할 수 있다. 두 헤더를 꼭 선언해야 한다.

```
#include <time.h>
#include <stdlib.h>

int main() {
        int n;
        srand(time(NULL));

        n = rand() % 3; // 0-3까지의 난수 생성
        return 0;
}
```

■ 엔터를 누르지 않고 사용자 입력 받기

int _getch(void): 화면에 출력 없이 키보드로부터 하나의 문자를 입력 받는 함수

- 엔터키를 누르지 않고도 키보드 입력을 받을 수 있고, 누른 키가 화면에 출력되지 않기 때문에 게임을 엔터키 없이 진행할 수 있다.
- 누른 키보드의 값을 리턴한다.
- conio.h 헤더에 포함되어 있다.

■ 반복문에서 키보드 입력 여부 확인하기

_getch()는 엔터키 없이 키보드의 입력을 받을 수 있지만, 키 입력이 들어오지 않을 시 다른 작업을 할 수 없다. 이를 해결하기 위해 키보드의 입력이 발생했을 때만 특정 행동을 할 수 있게 하는 _kbhit() 함수를 이용할 수 있다. (keyboard hit 함수)

- 키 입력이 들어올 시 이 함수는 true값을 반환한다.
- conio.h 헤더를 추가 한 후 이용할 수 있다.

키보드 입력이 들어왔을 때 조건 문으로 진입하고 특정 작업을 할 수 있다. 키보드 입력이 들어오지 않으면 반복문만 진행 될 뿐 아무 작업이 이루어지지 않는다. 본 코드의 예시는 키보드의 w키가 눌렸을 시 작업이 진행되는 코드이다.

```
while (1) {
    if (_kbhit())
    {
        keyInput = _getch();
        if (keyInput == 'w')
        {
         /* Statement */
        }
    }
}
```

■ 맵 관리 방법

맵은 char **map의 형태로 선언 한 뒤 2차원 동적 할당을 통해 선언한다.

- 맵은 오브젝트, 캐릭터에 대한 정보를 가지고 있다.
- 맵은 N x M (N=5, M=30)의 형태를 가지도록 한다.
- 출력 화면에 표시되는 단계 사이의 경계선 (--------) 의 경우는 맵 변수에 포함하지 않는 것이 관리하기 수월하다. (맵 출력 함수를 통해 출력 시에만 경계선을 출력)

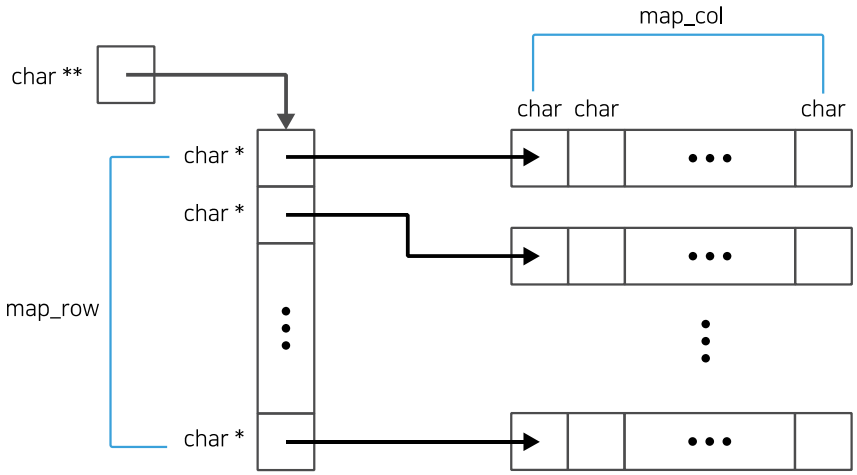
■ 2차원 배열의 동적할당

2차원 배열의 동적 할당은 다음의 과정을 통해 이루어진다.

- char ** 을 통해 메모리 공간의 첫번째 주소를 가리키는 변수를 선언한다.
- char 형태의 1차원 배열의 첫번째 주소를 가리키는 변수를 위해 char *로 이루어진 ROW개의 공간을 할당한다.
- ROW개의 char 형태의 1차원배열들을 각각 COL개 만큼 할당한다.

2차원 배열의 동적 할당

```
map = (char **)calloc(ROW, sizeof(char *));
for (i = 0; i < ROW; i++)
{
    map[i] = (char *)calloc(COL, sizeof(char));
}
```



■ **충돌 확인 방법**

매 프레임마다 또는 캐릭터가 움직일 때마다 캐릭터와 오브젝트의 충돌 검사를 해줘야 하며 이는 함수로 구현하면 많은 곳에서 쉽게 이용 할 수 있다. 충돌은 다음의 세가지 경우에서 발생 할 수 있으며, 조건문을 통해 각 상황에 대해서 충돌을 검사 한다.

- 캐릭터가 움직였을 때 나무와의 충돌을 감지 → 이동 불가능
- 캐릭터가 움직였을 때 자동차와의 충돌을 감지 → 게임 종료
- 캐릭터는 가만히 있지만, 자동차가 다가올 경우 충돌 감지 → 게임 종료

■ **과제에서 쓰면 좋은 변수/함수 관리 방법**

원활한 진행을 위해 몇가지 선언하면 좋을 변수들을 나열한다.

- int posX, posY: 캐릭터의 위치를 저장하는 변수
- char ** map: 맵을 위한 변수
- int * obsctacleType: i번째 나와야 할 물체들을 저장해놓은 변수
- int *obstacleNum: i 번째 물체의 개수를 저장해 놓은 변수 (위 변수와 2 차원으로 관리해도 좋다)
- 위 변수 외에도 다른 변수들도 많이 이용 될 수 있다.

구현하면 좋을 함수들을 나열한다.

- SetLevel(): 난이도를 설정하는 함수
- MenuDraw(): 메뉴창을 콘솔에 출력하는 함수
- MapInit(): 게임 시작 시 map을 초기화해주는 함수
  - 각 단계에 맞는 물체 생성

- 캐릭터 위치 초기화 및 생성
- 기타 게임에 필요한 변수들 초기화

- MapDraw(): 2차원 배열인 map을 이용하여 게임 화면에 맞게 출력하는 함수
- MapUpdate(): 매 프레임마다 불리는 함수로, 자동차가 한 칸씩 움직이도록 업데이트 하는 함수
- MapMove(): 캐릭터가 앞으로 나아갈 시 맵을 아래로 내리는 함수
- CharacterMove(): 사용자 입력을 받아 캐릭터를 움직이는 함수
- IsCollision(): 충돌을 판별하는 함수
- AddObejct() / 또는 AddCar(), AddTree(): 조건에 맞는 오브젝트를 만들어주는 함수
- 위 함수 외에도 많은 함수들이 이용 될 수 있다.

물론 위의 변수/함수를 이용하지 않아도 되며, 다른 변수/함수들을 이용해도 상관없다. 그러나 과제가 어려운 학생들은 위 변수/함수들을 이용하면 조금 더 수월하게 과제를 진행 할 수 있을 것으로 생각된다.