12. 상어 아일랜드 게임 만들기

2025 Fall

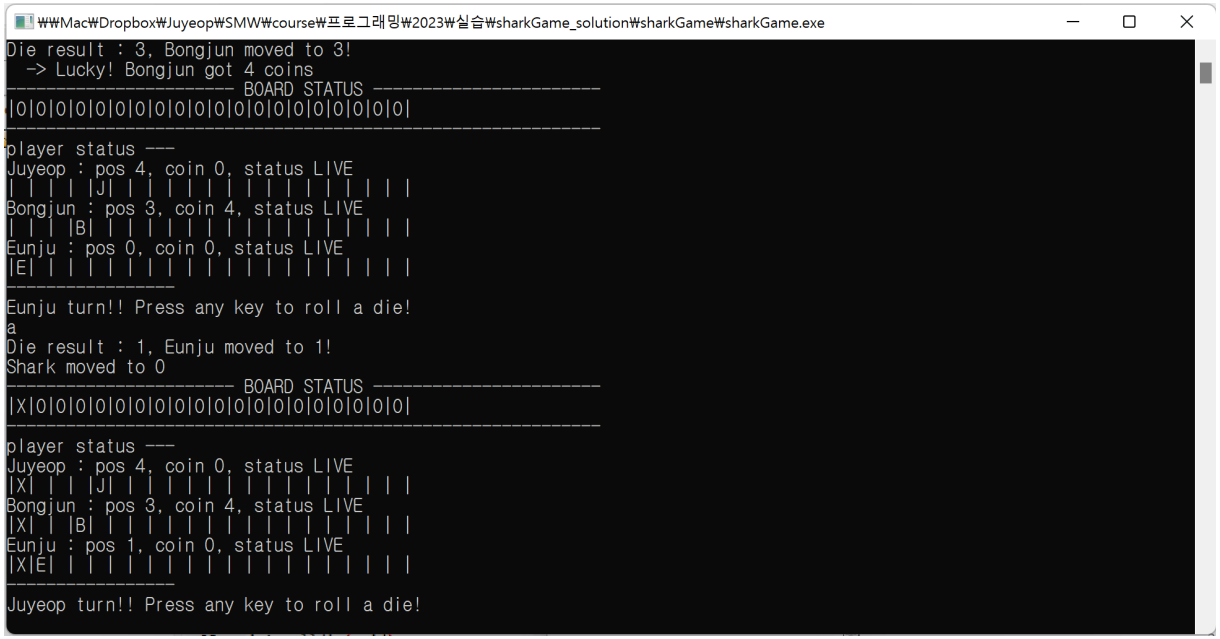
Objectives

- 지금까지 배운 문법을 활용하여 즐거운 상어 아일랜드 게임을 작성
  - 배열 등의 변수 문법을 활용하여 각종 요소들 표현
  - 일상적인 동작 규칙을 수식 및 코드로 표현
- 사이즈가 큰 프로그래밍에 대한 단계적 구현 방법 습득
  - 함수 모듈화 및 프로그램 구조화를 통해 전체를 세분화
  - 단계적 구현해나가는 과정에 대해 실습

게임 개요

- 우선 한 게임 해보죠.
  - sharkGame.exe 실행

https://brand.naver.com/koreaboardgames/



게임 규칙

- 입력된 플레이어 이름으로 표기 (플레이어수는 고정)
- 주사위를 굴리면서 앞으로 이동
    - 칸에 동전이 있으면 습득
- 상어가 쫓아오기 전에 끝까지 이동
- 살아있는 플레이어 중 동전이 가장 많은 플레이어가 승리

게임 흐름

- 각종 변수 초기화 및 플레이어 이름 입력
- 플레이어 별로 턴을 돌면서 반복 동작 수행
  - 게임 상태 출력
  - 주사위를 굴리고 이동 (동전 습득)
  - 적절한 시점에 상어 이동
- 게임 종료 후 출력

실습 1 : 구현 시작

- project 및 main.c 생성
- rand() 및 srand() 함수 사용을 위한 header include
  - #include <stdlib.h>
  - #include <time.h>
- main() 함수 윗부분에 rand 초기화 함수 호출
  - srand(unsigned(time(NULL));

실습 1 : 구현 시작

- 게임 시작 및 종료를 알리는 출력 꾸미기
  - 각자 개성에 맞게 (똑같이 안해도 됩니다.)

```
122 int main(int argc, char *argv[]) {
123
124     srand((unsigned)time(NULL));
125
126     printf("=====================================================\n");
127     printf("*****************************************************\n");
128     printf("                    BINGO GAME                       \n");
129     printf("*****************************************************\n");
130     printf("=====================================================\n\n\n");
131
132
133     printf("\n\n\n\n\n\n\n\n\n\n\n");
134     printf("=====================================================\n");
135     printf("*****************************************************\n");
136     printf("                  CONGRATULATION!!!!!                \n");
137     printf("*#&$*#^&*$&^@($!@(*   BINGO!!!!!  #)!@*(#)*%$(#*)@*(\n");
138     printf("                       YOU WIN!!!!!                  \n");
139     printf("*****************************************************\n");
140     printf("=====================================================\n\n");
141
142     return 0;
143 }
144
```

실습 2 : 설계

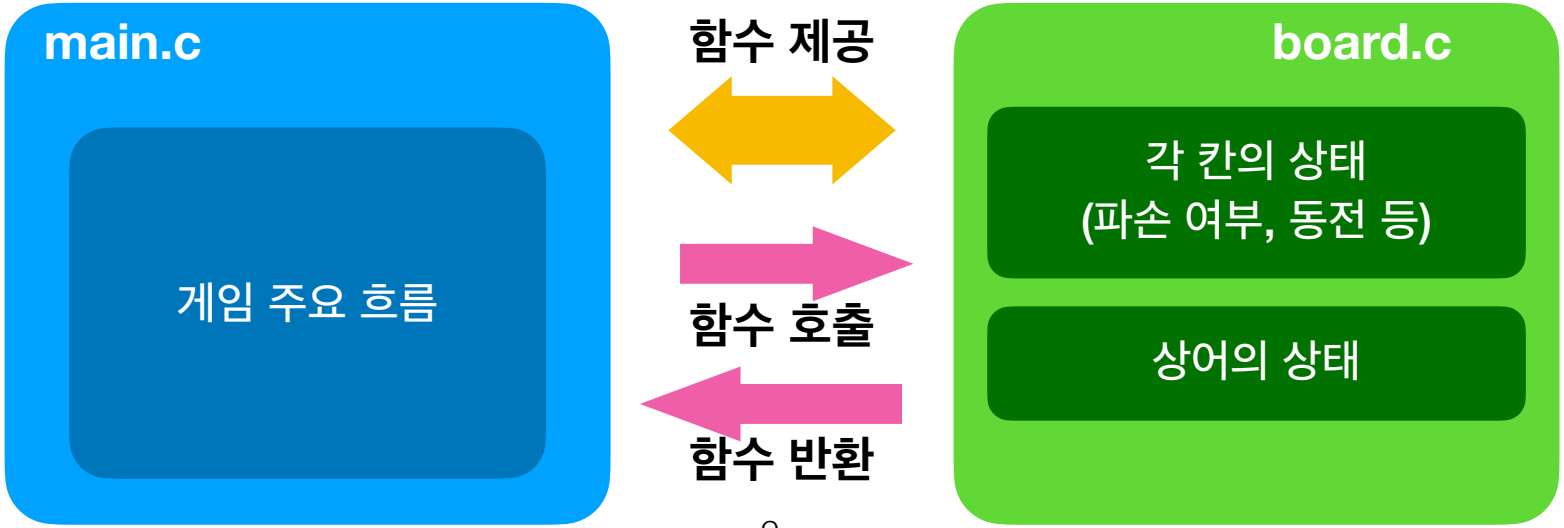
- 전체 동작 설계
  - main() 함수 수준에서의 큰 흐름에 대해 생각해보기
  - Pseudo-code로 sketch
    - 주석을 통해 동작의 흐름을 말로 표기해보기

```
void main() {
        //step 1. : ....
        //step 1-1. : ...

        //step 2. :
        .
        .
        .
}
```

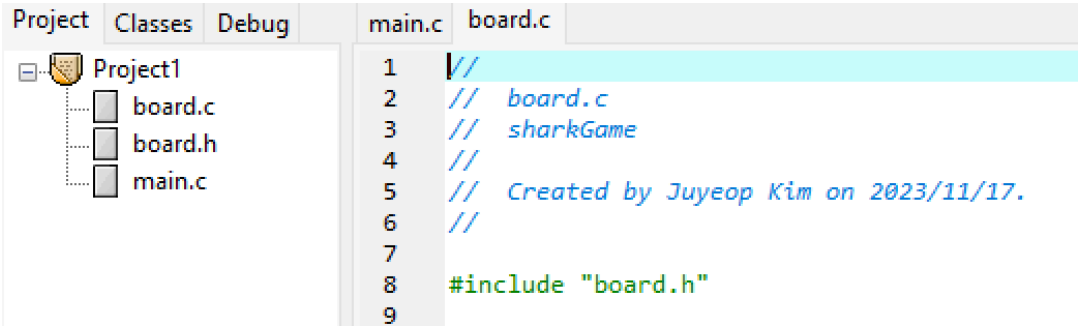
실습 2 : 설계

- 프로그램 구조 설계
  - 독립적인 모듈로 모을 수 있는 내용을 별도의 소스코드 파일에 담을 수 있도록 구조 잡기
- 예시 : 보드판 관련 동작 및 변수는 board.c에 담도록 설계



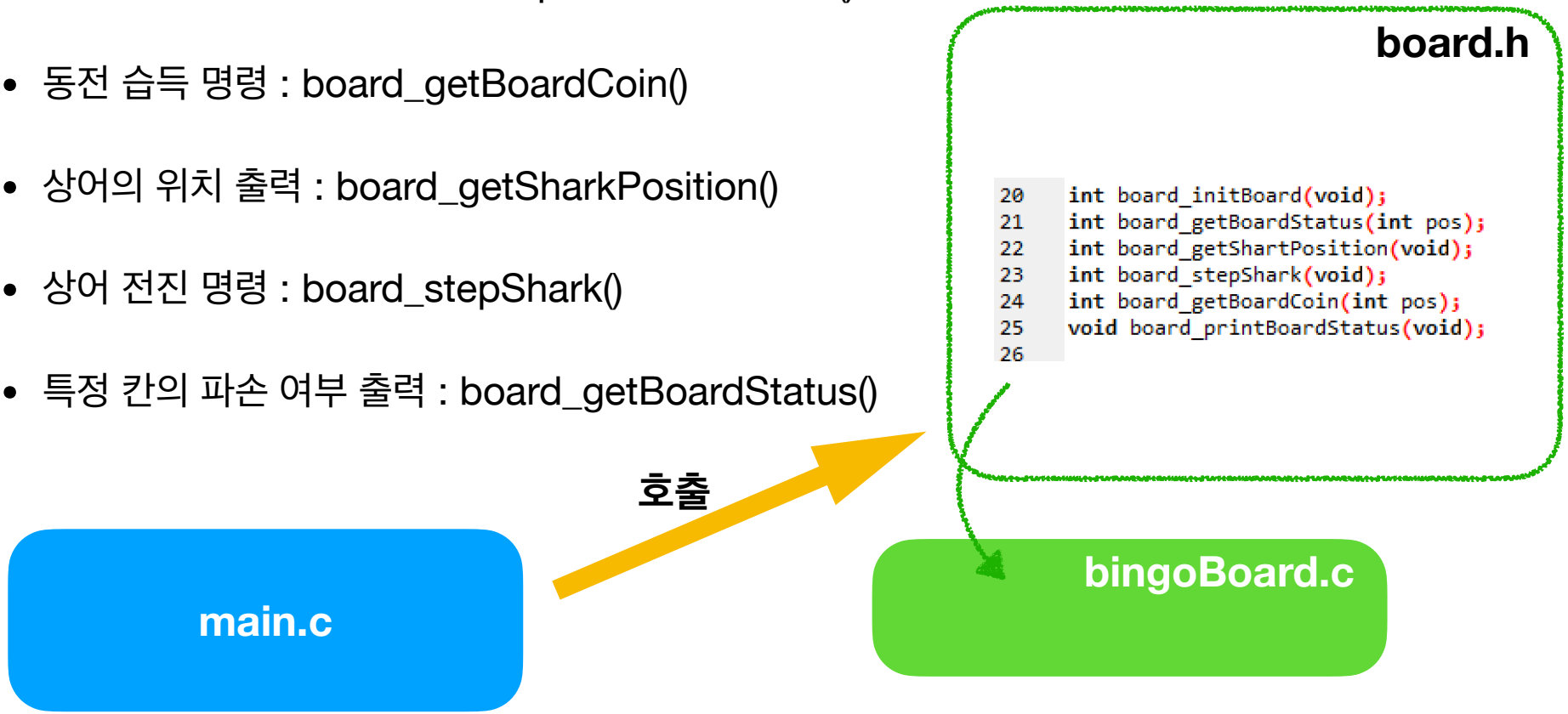
실습 2 : 전체 코드 설계

- board.c 및 board.h 두개 파일 추가
  - project에 새로운 파일을 추가
    - 새로운 파일에 stdio.h 및 stdlib.h를 include함
  - **새로 추가된 파일들이 project에 포함되어 있어야 같이 컴파일이 되면서 하나의 프로그램으로 생성된**



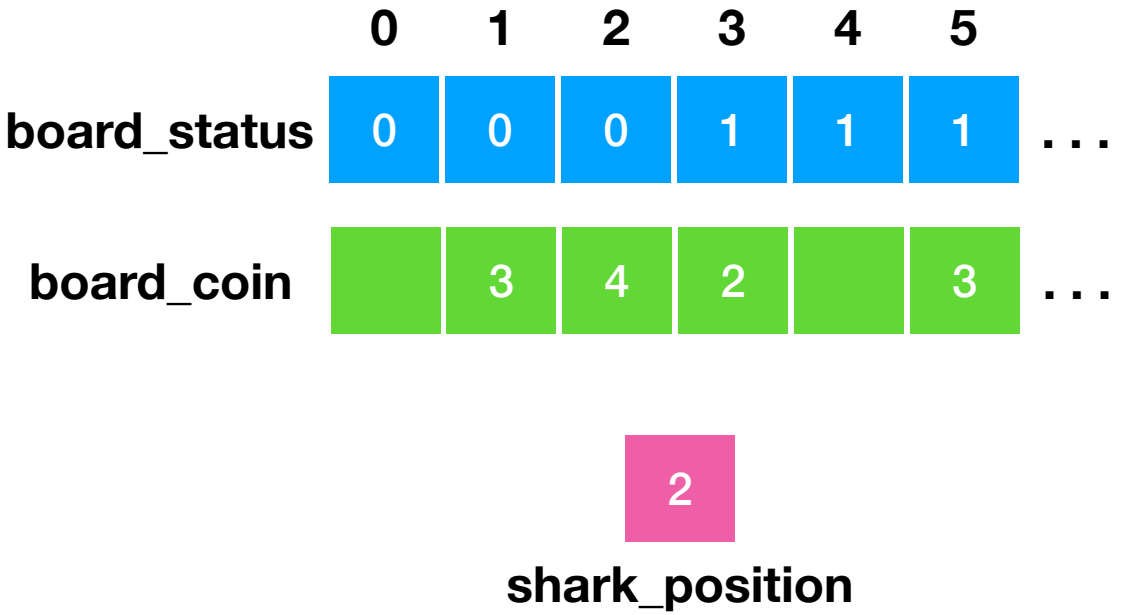
실습 2 : 전체 코드 설계

- 인터페이스 함수 설계 : 현재 생각나는 필요 함수를 대략적으로 (prototype 형태로) 정의
  - 보드 초기화 : board_initBoard()
  - 전체 보드의 상태 출력 : board_printBoardStatus()
  - 동전 습득 명령 : board_getBoardCoin()
  - 상어의 위치 출력 : board_getSharkPosition()
  - 상어 전진 명령 : board_stepShark()
  - 특정 칸의 파손 여부 출력 : board_getBoardStatus()



실습 3 : 게임보드 모델링

- Board : 1차원 배열들로 표현
  - 상어에 의한 파손 여부
  - 동전 존재 여부 및 동전 크기
- Board 관련 macro 정의
  - #define N_BOARD
  - #define BOARDSTATUS_OK 1
  - #define BOARDSTATUS_NOK 0
- 상어 : 위치를 저장한 정수형 변수



실습 3 : 게임보드 모델링

- void board_printBoardStatus(void)
  - 현재 보드판의 상태 출력
  - board_status 배열 값에 따라서 O 혹은 X 출력

```
19  void board_printBoardStatus(void)
20  {
21      int i;
22
23      printf("---------------------- BOARD STATUS ----------------------\n");
24      for (i=0;i<N_BOARD;i++)
25      {
26          printf("|");
27          if (board_status[i] == BOARDSTATUS_NOK)
28              printf("X");
29          else
30              printf("O");
31      }
32      printf("|\n");
33      printf("----------------------------------------------------------\n");
34  }
```

실습 3 : 게임보드 모델링

- void board_getBoardStatus(int pos)
  - board_status 배열 요소 반환
  - 내부 static 전역변수의 값을 main.c에 제공
- int board_getBoardCoin(int pos)
  - pos 번째 보드칸에 있는 동전을 줍는 동작
  - board_coin 배열의 값을 반환하면서 배열 값을 0으로 만듬

실습 3 : 게임보드 모델링

- int board_initBoard(void)
  - board_status 및 board_coin 초기화
  - board_coin에 임의로 동전 할당

MACRO를 정의하여 활용

```
#define N_COINPOS       12
#define MAX_COIN        4
```

```
int board_initBoard(void)
{
    for (i=0부터 N_BOARD까지)
        board_status의 i번째 요소 0 설정;
        board_coin의 i번째 요소 0 설정;
    //coin 할당
    for (i=0부터 N_COINPOS까지)
    {
        while (i번째 coin이 할당되지 않으면 반복)
        {
            동전을 놓을 칸을 랜덤으로 지정;
            if (칸에 동전이 이미 있지 않으면)
            {
                //i번째 coin 할당
                board_coin[지정칸] = (1~MAX_COIN 중 랜덤);
            }
        }
    }
}
```

실습 3 : 게임보드 모델링

- 구현 코드 검증
  - main함수에서 board.c의 함수들을 호출하면서 동작 확인

int pos=0; //위치 변수 선언
board_initBoard();

main 함수 내 do-while 반복문 배치

board_printBoardStatus();
주사위를 굴려서 이동 횟수 지정;
이동횟수만큼 pos 변경;
coinResult = board_getBoardCoin(pos);

```
#define MAX_DIE            6
|


int rolldie(void)
{
    return rand()%MAX_DIE+1;
}
```

실습 4 : 플레이어 모델링

- Macro 정의
  - N_PLAYER : 플레이어 수
  - PLAYER_STATUS_XXX : 플레이어 상태

```
#define N_PLAYER                4

#define PLAYERSTATUS_LIVE    0
#define PLAYERSTATUS_DIE     1
#define PLAYERSTATUS_END     2
```

- 플레이어 관련 변수 선언
  - 이름
  - 위치
  - 누적 동전 수
  - 현재 상태
  - 상태 설명 문자열

```
int player_position[N_PLAYER];
char player_name[N_PLAYER][MAX_CHARNAME];
int player_coin[N_PLAYER];
int player_status[N_PLAYER]; //0 - live, 1 - die, 2 - end
char player_statusString[3][MAX_CHARNAME] = {"LIVE", "DIE", "END"};
```

실습 4 : 플레이어 모델링

- 플레이어를 중심으로 게임의 흐름 파악
  - 플레이어가 필요한 동작 파악

1. Initialization
  1-1. board init.
  1-2. player init
2. game loop
  2-1. status printing
  2-2. roll die
  2-3. moving
  2-4. get coin
  2-5. check end & next turn

실습 4 : 플레이어 초기화

- 플레이어 상태 초기화
  - 각종 변수값 초기화 (position, coin, status)
  - 이름 입력

```
//step1-2 : initialize player
for (i=0;i<N_PLAYER;i++)
{
    player_position[i] = 0;
    player_coin[i] = 0;
    player_status[i] = PLAYERSTATUS_LIVE;
    printf("Player %i's name: ", i);
    scanf("%s", player_name[i]);
}
```

- 각 플레이어의 턴 동작 구현
  - game 반복문에서 한번 반복 후 다음 플레이어로 넘어감
  - turn 변수를 선언하고 반복마다 1씩 증가
    - N_PLAYER로 나누기연산 수행

```
int turn=0;
```

```
turn = (turn + 1)%N_PLAYER;//next turn
```

실습 4 : 플레이어 상태 출력

- 플레이어 상태 출력
  - printPlayerPosition(int player)
  - printPlayerStatus(void)

```c
// ----- EX. 4 : player ------------
void printPlayerPosition(int player)
{
    int i;

    for (i=0;i<N_BOARD;i++)
    {
        printf("|");
        if (player_position[player] == i)
            printf("%c", player_name[player][0]);
        else
        {
            if (board_getBoardStatus(i) == BOARDSTATUS_NOK)
                printf("X");
            else
                printf(" ");
        }
    }
    printf("|\n");
}

void printPlayerStatus(void)
{
    int i;
    printf("player status ---\n");
    for (i=0;i<N_PLAYER;i++)
    {
        printf("%s : pos %i, coin %i, status %s\n", player_name[i], player_position[i], player_coin[i],
            player_statusString[player_status[i]]);
        printPlayerPosition(i);
    }
    printf("-----------------\n");
}
```

실습 4 : 플레이어 이동

- 주사위 굴리기
  - 플레이어 턴 출력
  - 입력을 받고 주사위를 굴리는 함수 호출

```
printf("%s turn!! ", player_name[turn]);
printf("Press any key to roll a die!\n");
scanf("%d", &dum);
fflush(stdin);
dieResult = rolldie();
```

- 이동
  - player_position[turn] 값에 주사위 결과를 더함
    - 예외 상황에 대한 대처 (N_BOARD 이상 갔을 때)
  - 이동 결과 출력

실습 4 : 이동 결과 정리

- 동전 습득
  - 이동한 위치에서 board_getBoardCoin 함수 호출
  - 반환된 동전 값을 player_coin[turn]에 더함
  - 동전이 존재하는 경우 동전 습득 정보를 출력
- 보드 맨 끝까지 이동한 경우에 대한 처리
  - player_position[turn]이 N_BOARD-1인지 여부를 조건문으로 활용
  - player_status[turn]을 PLAYERSTATUS_END 값으로 설정

실습 4 : 플레이어 상태 별 턴 동작

- 플레이어 상태가 PLAYERSTATUS_LIVE인 경우만 반복문을 수행
  - continue 활용
  - 아래 코드를 while문의 적절한 위치에 삽입

```
if (player_status[turn] != PLAYERSTATUS_LIVE)
{
    turn = (turn + 1)%N_PLAYER;
    continue;
}
```

실습 4 : 검증

- while 문의 조건을 항상 참이 되도록 해서 동작 검증
  - 플레이어가 순서대로 턴을 도는지 여부
  - 각종 상태 출력 확인
  - 주사위에 따른 이동 동작
  - 이동 후 결과 확인 (동전 습득, 마지막 칸 이동 후 상태 등)

실습 5 : 상어 구현

- 플레이어가 모두 한번씩 턴을 돌면 상어가 동작
  - turn 변수가 0이 되는 조건을 활용해서 상어 동작 코드를 실행
  - board_stepShark() 함수 호출을 통해 상어의 이동 진행
  - checkDie() : 상어 이동 후 플레이어의 생존 여부 확인

```
if (turn == 0)
{
    int shark_pos = board_stepShark();
    printf("Shark moved to %i\n", shark_pos);
    //check die
    checkDie();
}
```

실습 5 : 상어 이동

```
#define MAX_SHARKSTEP    6
#define SHARK_INITPOS    -4
```

- board_stepShark()
  - 1 ~ MAX_SHARKSTEP 칸을 랜덤으로 이동
  - 이동하는 경로에서 board_status 배열요소 변경
  - shark_position값 갱신
- shark_position 초기화
  - board_initBoard함수의 적절한 위치에 SHARK_INITPOS값으로 초기화하는 코드 삽입

실습 5 : 플레이어 사망 여부

- checkDie()
  - 플레이어별로 for문으로 반복
    - player_position[i]의 board_status가 파손되었을때
      - player_status[i]를 PLAYERSTATUS_DIE로 변경
      - 사망 상태를 출력

실습 6 : 게임 종료

- do-while의 조건문 변경
  - game_end() : 게임 종료 조건이 만족되면 0 반환
    - 모든 플레이어가 PLAYERSTATUS_LIVE 상태가 아니면 종료
- 종료 후 생존 플레이어 수와 승자 플레이어 출력
  - getAlivePlayer() : 생존 플레이어 수 계산 및 반환
  - getWinner() : 생존 플레이어 중 동전이 가장 많은 플레이어 번호 반환

실습 6 : 게임 종료

- 구현 예시

```
int game_end(void)
{
    int i;
    int flag_end = 1;

    //if all the players are died?
    for (i=0;i<N_PLAYER;i++)
    {
        if (player_status[i] == PLAYERSTATUS_LIVE)
        {
            flag_end = 0;
            break;
        }
    }

    return flag_end;
}
```

```
int getAlivePlayer(void)
{
    int i;
    int cnt=0;
    for (i=0;i<N_PLAYER;i++)
    {
        if (player_status[i] == PLAYERSTATUS_END)
            cnt++;
    }

    return cnt;
}

int getWinner(void)
{
    int i;
    int winner=0;
    int max_coin=-1;

    for (i=0;i<N_PLAYER;i++)
    {
        if (player_coin[i] > max_coin)
        {
            max_coin = player_coin[i];
            winner = i;
        }
    }

    return winner;
}
```