문제 분석)

컴퓨터와 사용자가 각자의 주사위 숫자만큼 전진하며 땅에 도달하는 게임

1~20번 땅 중 랜덤으로 3개의 땅에는 황금이, 하나의 땅에는 폭탄이 숨겨져 있다.

황금이 숨겨진 땅에 도착한 사람은 100만원 어치의 황금을 얻는다.

5번의 ROUND 동안 황금이 숨겨진 땅에 더 많이 도달해 최종 금액이 더 큰 플레이어가 승리

단, 5 ROUND 안에 폭탄이 숨겨진 땅을 밟은 사람은 즉시 패배 & 게임 종료

컴퓨터가 굴린 주사위의 숫자는 랜덤 2) 사용자는 본인이 주사위의 숫자를 정해 입력한다. 3) 사용자가 입력한 숫자가 주사위의 숫자(1~6) 범위를 벗어난다면 다시 입력 받는다. 4) 주사위를 굴리는 순서는 사용자 → 컴퓨터의 순서가 되도록 한다.

그림과 같이 20개의 땅이 하나의 루프로 구성되어 있다. (1->2->….->19->20->1->2->…)

Ex. 19번 땅에 위치한 플레이어가 주사위를 굴려 5가 나왔다면 플레이어의 위치는 4

땅은 1부터 20까지 숫자로 넘버링 되어 있다.

황금이 숨겨져 있는 땅 3곳과 폭탄이 숨겨져 있는 땅 1곳의 위치는 랜덤으로 플레이어들은 모른다.

황금 땅, 폭탄 땅의 위치는 새로운 게임을 시작할 때마다 바뀐다.

황금이 숨겨져 있는 땅 3곳과 폭탄이 숨겨져 있는 땅 1곳의 위치는 서로 중복되지 않아야 한다.

이전 플레이어가 이미 도달한 적 있는 황금 땅이라 하더라도 황금은 계속해서 남아 있다

황금 땅에 도착하면 플레이어는 +100만원을 얻는다.

황금 땅에 도착할 때마다 플레이어가 가진 돈은 누적된다.

폭탄 땅에 도착한 플레이어는 게임에서 패배한다.

플레이어가 폭탄 땅을 밟으면 게임 (프로그램)은 종료된다.

게임은 최대 5라운드까지 진행된다.

각 라운드가 끝날 때 마다 결과 (플레이어들의 위치, 플레이어들 각각의 돈)를 보여준다.

의사 코드 및 소스 코드 분석)

set BOARD\_SIZE 20

set GOLD 1

set BOMB 2

YourTurn(data[], board[]) {

set dice to -1

while dice > 6 or dice < 1 {

if dice != -1 then

print "주사위의 수를 다시 입력해주세요.";

print "주사위의 수를 입력해주세요: ";

get dice;

}

if data[0] + dice >= BOARD\_SIZE then

set data[0] to data[0] + dice - BOARD\_SIZE;

else then

set data[0] to data[0] + dice;

print "당신은 %d번 땅에 도착했습니다.", data[0];

if board[data[0]] == GOLD then {

set data[1] to data[1] + 100;

print "황금을 찾았습니다! +100만원";

print "보유 금액: 0 -> %d 만원", data[1];

} else if board[data[0]] == BOMB then {

print "당신은 폭탄에 맞았습니다.";

print "Game Over.";

return 1

}

return 0

}

ComputerTurn(data[], board[]) {

set dice to rand() % 6 + 1;

print “The computer rolled a %d!", dice;

if data[0] + dice >= BOARD\_SIZE then

set data[0] to data[0] + dice - BOARD\_SIZE;

else then

set data[0] to data[0] + dice;

print "Computer은 %d번 땅에 도착했습니다.", data[0];);

if board[data[0]] == GOLD then {

set data[1] to data[1] + 100;

print "황금을 찾았습니다! +100만원";

print "보유 금액: 0 -> %d 만원", data[1];

} else if board[data[0]] == BOMB then {

print "당신은 폭탄에 맞았습니다.";

print "Game Over.";

return 1

}

return 0

}

main() {

print "중간프로젝트 Problem 3";

print "학 번:202320000 , 이름: 김00";

print "============= 황금 땅 찾기 게임 =============“;

print " 게임 진행 중 황금 땅에 도착하면 +100만원 지급";

print " 5 라운드 후에 보유한 금액이 더 많은 플레이어가 승리합니다.";

set board[BOARD\_SIZE] to 0 ;

set card[4] to 0;

for i = 0; i < 4; i++ {

set card[i] to (rand() % BOARD\_SIZE) + 1;

for j = 0; j < i; j++

if card[i] == card[j] then

i--

}

for i = 0; i < 3; i++

set board[card[i]] to GOLD;

set board[card[3]] to BOMB;

set data\_player[2] to 0;

set data\_computer[2] to 0;

set round to 1;

set winner to 0;

while round <= 5 {

print "============ ROUND %d ===========", round;

print "[Your turn]";

set reward\_player to YourTurn(data\_player, board);

if reward\_player then {

set winner to -2;

break

}

Sleep(1000)

print "[Computer's turn]";

set reward\_computer to ComputerTurn(data\_computer, board);

if reward\_computer then {

winner = -1;

break;

}

Sleep(1000)

print "=================================";

print "========== %d ROUND 결과 =========", round++;

print "- Your position: %d번 땅, money: %d 만원", data\_player[0], data\_player[1];

print "- Computer's position: %d번 땅, money: %d 만원", data\_computer[0], data\_computer[1];

}

if winner == -1 then {

print "You Win!";

} else if winner == -2 then {

print "Computer Wins!";

} else then {

print "================= GAME OVER ================";

print " Total money: Computer %d 만원 vs. You %d 만원\n", data\_computer[1], data\_player[1];

if data\_player[1] > data\_computer[1] then {

print "You Win!";

} else if data\_player[1] < data\_computer[1] then {

print "Computer Wins!";

} else then

print "무승부 입니다.";

}

return 0

}

int YourTurn(int data[], int board[])

data[] 변수는 [0]번 인덱스에는 사용자의 위치를, [1]번 인덱스에는 사용자의 현재 황금 수량을 저장한 배열이며, board[]는 모든 땅의 정보를 저장한 배열이다.

사용자의 차례에 사용자 프로그램을 수행하는 함수. 사용자의 턴이 시작되면 주사위 변수를 -1에 저장한다.

그리고 올바른 입력을 받을 때까지 계속 입력을 받는데, 이 때 최초 초기화한 주사위값 -1이 아닌 경우에만 다시 입력하라는 메세지를 출력한다.

올바른 주사위 값을 받은 이후, 현재 칸에서 이동할 칸을 더한 값이 땅 사이즈 20을 넘을 경우, 해당 값에 20을 빼준다. (ex: 18 + 5 = 23 -> 23 - 20 = 3) 그렇지 않을 경우 이동할 칸만큼 이동하며, 이동이 완료되면 도착한 땅의 번호를 출력한다.

이후 도착한 땅이 황금 또는 폭탄인 경우에 대해서 처리하는데, 보드 위 좌표가 황금일 경우, 보유 황금을 100 증가시키고, 황금을 찾았다는 메세지와 현재 보유금액에 대한 출력을 진행한다. 보드 위 좌표가 폭탄일 경우, 바로 게임을 종료해야하며 1을 리턴하여 함수를 종료한다. 그 이외의 경우 0을 리턴하여 함수를 종료한다.

int ComputerTurn(int data[], int board[])

data[] 변수는 [0]번 인덱스에는 컴퓨터의 위치를, [1]번 인덱스에는 컴퓨터의 현재 황금 수량을 저장한 배열이며, board[]는 모든 땅의 정보를 저장한 배열이다.

컴퓨터의 차례에 사용자 프로그램을 수행하는 함수. 컴퓨터의 턴이 시작되면 주사위 변수에 난수 생성을 통하여 1~6사이의 정수를 저장하고, 컴퓨터가 주사위를 굴렸다는 출력을 한다.

사용자 차례 함수와 마찬가지로, 현재 칸에서 이동할 칸을 더한 값이 땅 사이즈 20을 넘을 경우, 해당 값에 20을 빼준다. (ex: 18 + 5 = 23 -> 23 - 20 = 3) 그렇지 않을 경우 이동할 칸만큼 이동하며, 이동이 완료되면 도착한 땅의 번호를 출력한다.

이후 도착한 땅이 황금 또는 폭탄인 경우에 대해서 처리하는데, 보드 위 좌표가 황금일 경우, 보유 황금을 100 증가시키고, 황금을 찾았다는 메세지와 현재 보유금액에 대한 출력을 진행한다. 보드 위 좌표가 폭탄일 경우, 바로 게임을 종료해야하며 1을 리턴하여 함수를 종료한다. 그 이외의 경우 0을 리턴하여 함수를 종료한다.

int main(void)

프로그램의 전반적인 흐름을 메인함수에서 진핸한다. 프로젝트 제목과 학생의 정보, 게임의 정보에 대한 출력을 실시한다.

board[] 변수는 땅의 크기만큼 모든 땅을 0으로 초기화하고, car[] 변수는 [0], [1], [2]번 인덱스에는 황금을, [3]번 인덱스에는 폭탄을 저장한다. 랜덤 시드를 생성하고, 황금 1, 2, 3과 폭타의 순서로 난수를 생성하되, 이전에 생성된 난수들과 비교하여 중복되지 않도록 생성하며, 중복될 경우, 인덱스를 다시 1 줄이는 방식으로 난수를 생성한다. 이 후 보드에 3개의 황금과 폭탄의 인덱스 값에 GOLD와 BOMB으로 표시한다.

이후 플레이어와 컴퓨터의 데이터를 저장할 변수 data\_player, data\_computer을 생성하고, 라운드와 승자를 나타낼 변수를 만든다.

while 반복문을 통해 1라운드 부터 5라운드까지만 진행하며, 플레이어 -> 컴퓨터 순으로 차례를 진행하고, 각 라운드가 끝날 때마다 위치와 보유 황금에 대한 정보를 출력한다. 플레이어의 턴을 진행하는 함수를 실행하고, 만약 1이 반환된다면, 플레이어의 차례에 플레이어가 폭탄을 밟은 것으로 처리하여 즉시 반복문을 종료하고 승리자에 대한 출력을 진행하고 게임을 종료한다. 마찬가지로 컴퓨터의 차례를 진행하고 함수가 1을 반환할 경우, 반복문을 종료하고 승리자에 대한 출력을 진행한다.

폭탄을 밟지 않고 정상적으로 게임이 종료되었다면, 5라운드 종료 이후 황금 보유 상황을 비교하여, 승패에 대한 여부를 출력해준다.

결과)