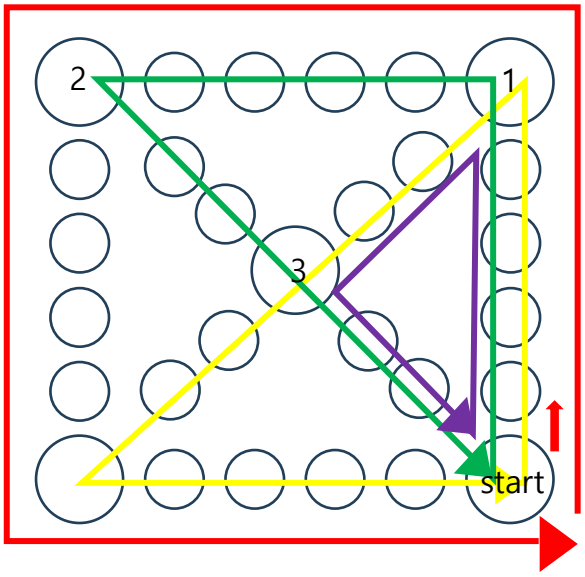
2024-1 융합 프로그래밍 중간프로젝트

윷놀이 구현

아주대학교 고준환(jhkoh2000@ajou.ac.kr)

AJOU UNIV.
1973

Problem – 윷놀이 구현



1. 문제 설명

1) 왼쪽 그림과 같은 말판에서 컴퓨터와 윷놀이를 진행.
2) start 칸에서 출발하여 반시계 방향으로 돌아 start까지 돌아오면 승리.
3) 도-1칸, 개-2칸, 걸-3칸, 윷-4칸, 모-5칸 이동. 빽도의 경우 뒤로 한칸 이동.
4) 그림에 표기된 1, 2, 3 칸에 있을시 지름길을 필수로 사용.
5) 상대방의 말을 잡을 수 있으며, 잡힌 말은 다시 start에서 시작.
6) 컴퓨터와 한 턴씩 돌아가면서 게임을 진행.
7) 윷/모가 나왔을 때와 상대방 말을 잡았을 때는 윷을 한번더 던질 수 있음.
8) 윷/모가 나와 윷을 한번 더 던지는 경우, 던질 수 있는 횟수만큼 윷을 던지고 그 총합만큼만 이동할 수 있음. (예시: 윷/모/개가 나왔을 경우, 11칸을 한번에 이동함.)
9) 시작점에서 빽도가 나오는 경우는 고려하지 않음.

Problem – 윷놀이 구현

2. 문제 조건

① 윷 던지기

1) 사용자/컴퓨터가 던져서 나온 결과는 랜덤

2) 윷을 던져서 나온 결과는 빽도, 도, 개, 걸, 윷, 모의 6개 경우만 존재

3) 윷과 모가 나온 경우는 한번 더 윷을 던짐.

4) 윷을 던지는 순서는 사용자 → 컴퓨터의 순서가 되도록 한다.

5) 윷과 모가 나와 연속으로 던지는 경우, 던져서 나온 총합만큼 한번에 말을 이동시킨다.

Problem – 윷놀이 구현

② 말판의 조건

1) 그림과 같이 지름길을 이용하지 않을 시 총 20칸을 이동해야 완주.

2) 지름길을 한 번 이용 시 총 16칸, 두 번 이용할 시 총 11칸을 이동해야 완주.

3) Start 칸에서 시작해서 Start 칸으로 먼저 돌아오면 승리.

4) Start 칸에서 빽도가 나왔을 때는 말을 움직이지 않는다.

Problem – 윷놀이 구현

③ 게임 방식

1) 컴퓨터와 사용자는 각각 한 개의 말만으로 게임을 진행한다.

2) 컴퓨터와 사용자가 윷을 던질 때, 사용자로부터 1을 입력받도록 한다.

3) 상대방과 같은 위치에 갔을 경우, 상대방 말을 잡을 수 있으며, 이 때 잡힌 말은 Start 칸에서 다시 시작하게 되고, 잡은 말은 윷을 한번 더 던진다.

4) 지름길을 이용했을 경우, 지름길을 이용한다는 메시지를 출력하며, 현재까지 몇번의 지름길을 이용했는지 출력한다.

5) 턴이 끝날 때마다, 지금까지 총 몇 칸을 이동하였는지와 완주까지 몇 칸을 이동해야 하는지 메시지를 출력한다.

6) 윷을 다 던지고 말을 이동할 때, 몇 칸을 이동한지 메시지를 출력한다.

Problem – 윷놀이 구현

3. 구현 조건

1) 윷을 던지게 하는 **함수**를 **재귀함수**로 구현.

2) 윷을 던지게 할 때, **난수 발생 함수**로 구현.

3) 지름길을 이용가능한 위치에 있는지 판단하는 **함수**를 구현

4) 지름길을 이용할 때, 앞으로 가야할 칸수를 줄여주는 **함수**를 구현.

5) 지름길 이용가능한 위치에서 빽도가 나왔을 때, 이미 줄어든 앞으로 가야할 칸수를 다시 복구시켜 주는 **함수**를 구현.

6) 던진 윷만큼 이동했을 때, 상대방 말을 잡을 수 있는지 판단하는 **함수**를 구현.

7) 한 턴에서 총 몇 칸을 이동할 수 있는지 저장하는 **전역변수** 사용.

Problem – 윷놀이 구현

4. 결과 화면 제출

1) 아래 예시로 출력된 결과 창 화면 모두 출력

2) 학번, 이름 콘솔 창 출력 (**없을 시, copy 처리**)

3) 예시와 동일하게 출력되도록 printf문 작성 (띄어쓰기, 줄 바꿈, 문구)

Problem – 윷놀이 구현

5. 출력 예시

1) 게임 시작 화면

```
중간 프로젝트
학번: 2024xxxxx, 이름: 고준환

============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요!
```

2) 1을 입력하여 윷을 던진 후의 화면

```
중간 프로젝트
학번: 2024xxxxx, 이름: 고준환

============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요! 1
걸이 나왔습니다!
이동할 수 있는 칸수는 3칸 입니다!
당신 말의 이동한 칸수는 3 입니다.
완주까지 17 칸 남았습니다.
============================================================
컴퓨터의 차례입니다!
윷을 던지게 하려면 1을 입력하세요!
```

Problem – 윷놀이 구현

3) 모 또는 윷이 나왔을 때의 화면

```
중간 프로젝트
학번: 2024xxxxx, 이름: 고준환

============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요! 1
모가 나왔습니다! 1을 입력하면 한번 더 던집니다!
```

```
중간 프로젝트
학번: 2024xxxxx, 이름: 고준환

============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요! 1
모가 나왔습니다! 1을 입력하면 한번 더 던집니다! 1
빽도가 나왔습니다!
이동할 수 있는 칸수는 4칸 입니다!
당신 말의 이동한 칸수는 4 입니다.
완주까지 16 칸 남았습니다.
============================================================
컴퓨터의 차례입니다!
윷을 던지게 하려면 1을 입력하세요!
```

4) 시작점에서 빽도가 나왔을 때의 화면

```
중간 프로젝트
학번: 2024xxxxx, 이름: 고준환

============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요! 1
빽도가 나왔습니다!
이동할 수 있는 칸수는 -1칸 입니다!
당신 말의 이동한 칸수는 0 입니다.
완주까지 20 칸 남았습니다.
============================================================
컴퓨터의 차례입니다!
윷을 던지게 하려면 1을 입력하세요!
```

Problem – 윷놀이 구현

4) 지름길을 이용할 때의 화면

```
중간 프로젝트
학번: 2024xxxxx, 이름: 고준환

============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요! 1
빽도가 나왔습니다!
이동할 수 있는 칸수는 -1칸 입니다!
당신 말의 이동한 칸수는 0 입니다.
완주까지 20 칸 남았습니다.
============================================================
컴퓨터의 차례입니다!
윷을 던지게 하려면 1을 입력하세요! 1
윷이 나왔습니다! 1을 입력하면 한번 더 던집니다! 1
도가 나왔습니다!
이동할 수 있는 칸수는 5칸 입니다!
컴퓨터 말이 이동한 칸수는 5 입니다.
완주까지 15 칸 남았습니다.
============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요! 1
윷이 나왔습니다! 1을 입력하면 한번 더 던집니다! 1
모가 나왔습니다! 1을 입력하면 한번 더 던집니다! 1
개가 나왔습니다!
이동할 수 있는 칸수는 11칸 입니다!
당신 말의 이동한 칸수는 11 입니다.
완주까지 9 칸 남았습니다.
============================================================
컴퓨터의 차례입니다!
윷을 던지게 하려면 1을 입력하세요! 1
개가 나왔습니다!
이동할 수 있는 칸수는 2칸 입니다!
지름길을 이용합니다!
현재 지름길 이용횟수: 1
컴퓨터 말이 이동한 칸수는 7 입니다.
완주까지 9 칸 남았습니다.
============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요!
```

5) 지름길 이용가능 지점에서 빽도가 나왔을 때의 화면

```
중간 프로젝트
학번: 2024xxxxx, 이름: 고준환

============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요! 1
윷이 나왔습니다! 1을 입력하면 한번 더 던집니다! 1
도가 나왔습니다!
이동할 수 있는 칸수는 5칸 입니다!
당신 말의 이동한 칸수는 5 입니다.
완주까지 15 칸 남았습니다.
============================================================
컴퓨터의 차례입니다!
윷을 던지게 하려면 1을 입력하세요! 1
모가 나왔습니다! 1을 입력하면 한번 더 던집니다! 1
개가 나왔습니다!
이동할 수 있는 칸수는 7칸 입니다!
컴퓨터 말이 이동한 칸수는 7 입니다.
완주까지 13 칸 남았습니다.
============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요! 1
빽도가 나왔습니다!
이동할 수 있는 칸수는 -1칸 입니다!
당신 말의 이동한 칸수는 4 입니다.
완주까지 16 칸 남았습니다.
============================================================
컴퓨터의 차례입니다!
윷을 던지게 하려면 1을 입력하세요!
```

Problem – 윷놀이 구현

6) 사용자 말이 잡혔을 때의 화면

```
중간 프로젝트
학번: 2024xxxxx, 이름: 고준환

============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요! 1
걸이 나왔습니다!
이동할 수 있는 칸수는 3칸 입니다!
당신 말의 이동한 칸수는 3 입니다.
완주까지 17 칸 남았습니다.
============================================================
컴퓨터의 차례입니다!
윷을 던지게 하려면 1을 입력하세요! 1
걸이 나왔습니다!
이동할 수 있는 칸수는 3칸 입니다!
당신 말이 잡혔습니다! 컴퓨터가 윷을 한번 더 던집니다!
컴퓨터의 차례입니다!
윷을 던지게 하려면 1을 입력하세요!
```

7) 컴퓨터 말을 잡았을 때의 화면

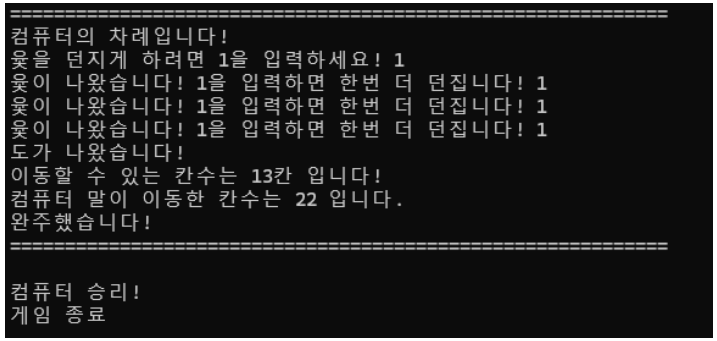
```
중간 프로젝트
학번: 2024xxxxx, 이름: 고준환

============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요! 1
빽도가 나왔습니다!
이동할 수 있는 칸수는 -1칸 입니다!
당신 말의 이동한 칸수는 0 입니다.
완주까지 20 칸 남았습니다.
============================================================
컴퓨터의 차례입니다!
윷을 던지게 하려면 1을 입력하세요! 1
빽도가 나왔습니다!
이동할 수 있는 칸수는 -1칸 입니다!
컴퓨터 말이 이동한 칸수는 0 입니다.
완주까지 20 칸 남았습니다.
============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요! 1
빽도가 나왔습니다!
이동할 수 있는 칸수는 -1칸 입니다!
당신 말의 이동한 칸수는 0 입니다.
완주까지 20 칸 남았습니다.
============================================================
컴퓨터의 차례입니다!
윷을 던지게 하려면 1을 입력하세요! 1
윷이 나왔습니다! 1을 입력하면 한번 더 던집니다! 1
도가 나왔습니다!
이동할 수 있는 칸수는 5칸 입니다!
컴퓨터 말이 이동한 칸수는 5 입니다.
완주까지 15 칸 남았습니다.
============================================================
당신의 차례입니다!
윷을 던지려면 1을 입력하세요! 1
윷이 나왔습니다! 1을 입력하면 한번 더 던집니다! 1
도가 나왔습니다!
이동할 수 있는 칸수는 5칸 입니다!
당신 말의 이동한 칸수는 5 입니다.
상대방 말을 잡았습니다! 윷을 한 번 더 던집니다!
당신의 차례입니다!
윷을 던지려면 1을 입력하세요!
```

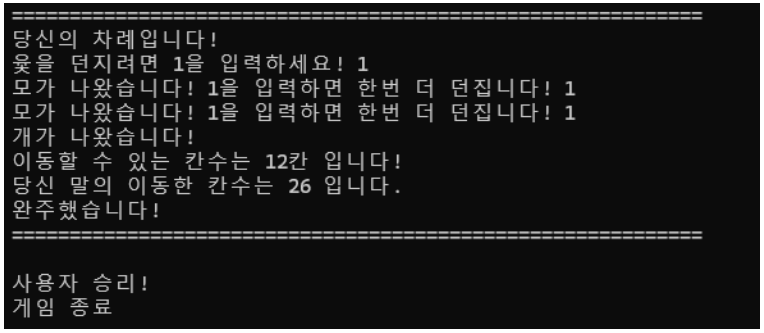
Problem – 윷놀이 구현

8) 먼저 한 바퀴를 완주하는 플레이어가 승리

1. 컴퓨터가 승리



2. 사용자가 승리



Problem – 윷놀이 구현

6. 성적 평가

- **코드(60%)**

1) roll() 구현 10% -> 윷을 던지는 함수

- 랜덤 함수 구현 5%
- 재귀함수 구현 5%

2) is_shortcut_available() 구현 5% ->현재 위치에서 지름길 사용이 가능한지 판별하는 함수

- 조건문 구현 5%

3) using_shortcut() 구현 5% -> 현재 위치에서 지름길 사용이 가능할 때, 지름길을 이용하는 함수

- 조건문 구현 5%

Problem – 윷놀이 구현

6. 성적 평가

- **코드(60%)**

4) back_shortcut() 구현 5% -> 지름길 사용 가능한 위치에서 빽도가 나왔을 때, 이미 줄어든 가야 할 칸수를 다시 보상해주는 함수

- 조건문 구현 5%

5) is_catch_available() 구현 5% -> 윷이 나온만큼 이동했을 때, 상대방 말을 잡을 수 있는 상태인지 판별하는 함수

- 조건문 구현 5%

Problem – 윷놀이 구현

3) main 함수 구현 30%

- 반복문을 활용하여 사용자/컴퓨터 번갈아 진행하는 턴제 게임 구현 10%
- 지름길 이용 구현 10%
- 출력문구 구현 5%
- 상대방 말 잡는 행동 구현 5%

[유의사항]

- 컴파일 에러 시 **코드 점수 0점**
- C파일이 아닌 다른 파일 제출시, **코드 점수 0점**
- **주석 없을 시, 코드 점수 0점**
- 모든 입력에 대해서 결과가 올바르게 출력되어야 함. 특정 입력에만 만족하게 구현할 시, 해당 항목 0점

Problem – 윷놀이 구현

6. 성적 평가

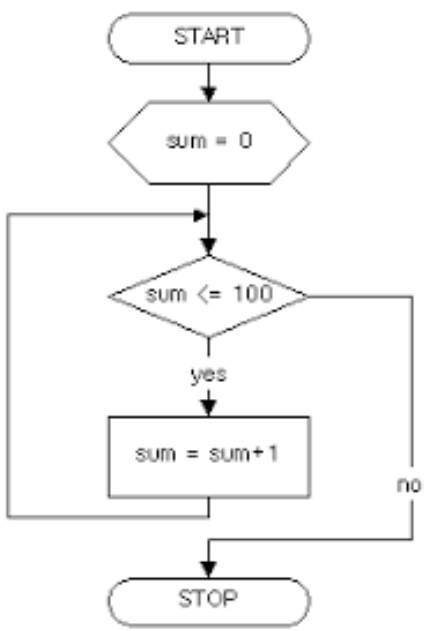
- **보고서(40%)**
  - 문제분석 : 문제를 재정의 후 제약사항, 요구사항, 입력, 결과 등 정리 (10%)
  - 순서도 및 pseudo-code: 순서도 혹은 pseudo-code 기재 및 설명 (10%)
  - 소스코드 분석 : 구현함수의 입출력 구조 및 기능 설명 (10%)
    (코드 캡쳐 후 분석)
  - 결과화면 분석 : 결과화면 캡쳐 후 과정 구현 확인, 모든 기능 설명 (10%)
- **기타**
  - **Copy 적발 시, 해당 프로젝트 0점 처리.**

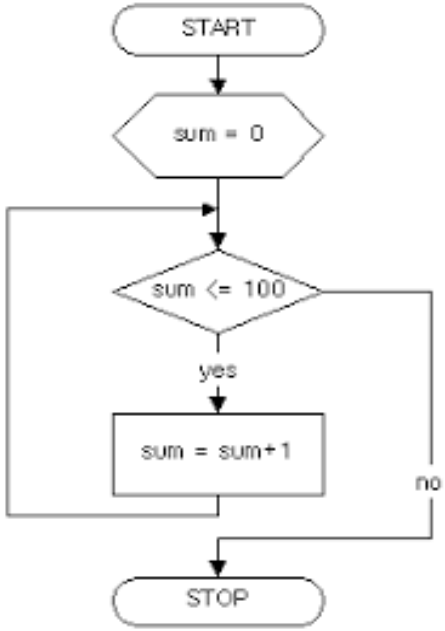
순서도(flowchart)로 알고리즘 기술

- 순서도(flowchart) :
  어떠한 일을 처리하는 과정을 순서대로 간단한 기호와 도형으로 도식화한 것

- 순서도 기호 및 사용법: draw.io

기호	명칭	용도
	흐름선	작업의 흐름 명시
	단말	순서도의 시작, 끝을 나타냄
	처리	처리작업 명시
	입출력	데이터 입력, 출력
	수동입력	키보드를 이용한 입력
	준비	사전 준비과정
	판단	명시된 조건 비교 판단
	연결자	같은 페이지 순서도간 연결
	페이지 연결자	다른 페이지내 있는 순서도 연결 (기호안에 명칭기재)
	결합	기존의 흐름선에 합류
	설명문	지정된 부분 설명





의사코드(pseudo-code)로 알고리즘 기술

- 의사코드(pseudo-code)
  - 프로그램을 작성할 때 각 모듈이 작동하는 논리를 표현하기 위한 언어
  - 일반적인 언어로 코드를 흉내 내어 알고리즘을 써놓은 코드

- 의사코드 표기법

Type of operation	Symbol	Example
Assignment	← or :=	$c \leftarrow 2\pi r$, $c := 2\pi r$
Comparison	=, ≠, <, >, ≤, ≥	
Arithmetic	+, −, ×, /, mod	
Floor/ceiling	⌊, ⌋, ⌈, ⌉	$a \leftarrow \lfloor b \rfloor + \lceil c \rceil$
Logical	**and**, **or**	
Sums, products	∑ ∏	$h \leftarrow \sum_{a \in A} 1/a$

```
ArrayMax(A,n)

  tmp ← A[0];
  for i←1 to n-1 do
          if tmp < A[i] then
                  tmp ← A[i];
  return tmp;
```