2021년 2학기

**프로그래밍과 문제해결**

**Assignment #1**

담당교수: 윤은영

학번: 20210370

학과: 무은재학부

이름 권민경

POVID ID: mkkwon

명예서약(Honor Code)

“나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.”

**주루마블**

1. **문제의 개요**

본 프로그램을 간략히 설명하면 다음과 같다.

* 부루마블을 술자리에서 즐기기 위해 변형 시킨 게임이다.
* USER는 매 턴 주사위를 던지고 주사위 값은 랜덤으로 지정된다.
* COMPUTER는 주사위를 던지지 않으며 보트 칸에 따라 술을 마신다.
* 먼저 14잔을 마시는 PLAYER가 패한다.

입력부: 게임을 진행하기 위한 명령어를 숫자로 입력하며, 사용자가 주사위를 굴리고 각 보드 칸에서 지시하는 내용을 입력한다.

처리부: 입력 받은 값에 따라 게임이 진행되며, 현재 보드 위치를 map계산, 현재 마시는 주량, 누적 주량을 계속 계산한다.

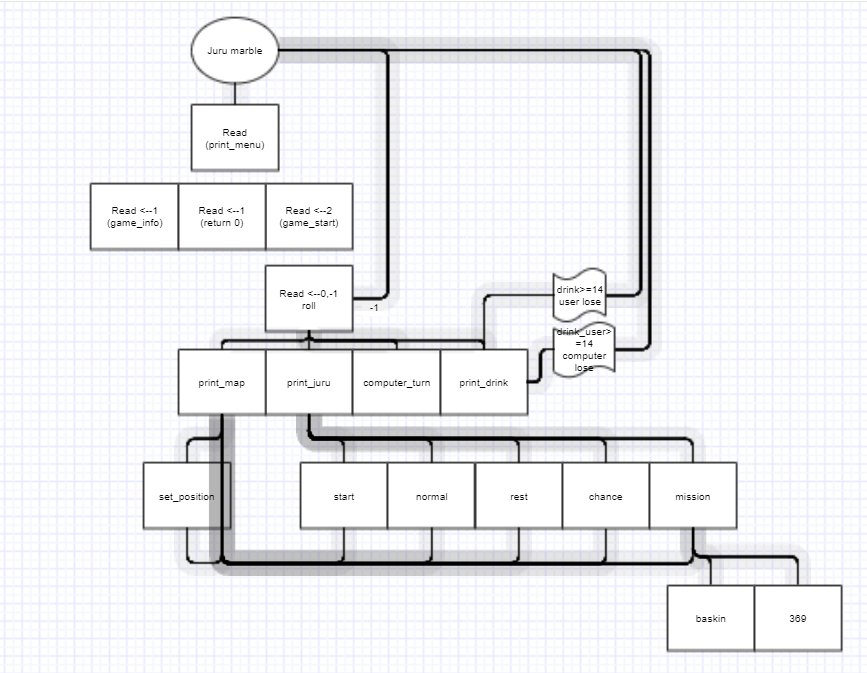
출력부: map 위치, 현재 마시는 잔 수, 누적 주량을 출력한다.

* Map: 사용자의 현재 위치를 나타내는 함수; set position 함수를 불러온다.
* Drink: 현재 누적 잔 수를 출력한다. 만약 누적 잔 수 가 14일 경우 게임을 종료한다.

1. **알고리즘**

본 프로그램 작성을 위한 알고리즘을 Pseudo 코드 형태로 나타내면 다음과 같다.

|  |
| --- |
| Pseudo-algorithm for juru marble |
| //변수는 선언된 것으로 가정   1. Include essential headers <stdio.h>, <stlib.h> and <time.h> 2. print menu 3. promote user, printing game info and game start 4. Promote user, printing map, drink and play\_juru function  * Printing map: using ‘ ’(space) and ‘O’ to indicate the user’s position * Drink: using ‘for’ phrase  1. For play\_juru, five functions must be working  * Start: drink/drink\_comp++ * Normal : drink++/drink\_comp++ * Rest: drink/drink\_comp++ * Chance: either drink-=2 or drink+=3 * Mission: either drink+=4 or drink\_comp+=4  1. For the result   When user lose, computer does not drink |

위의 의사 알고리즘을 flow chart를 통해 표현하면 아래와 같다.

1. **프로그램 구조 및 설명**
2. 메뉴를 프린트 하는 값 입력

* 1. 게임설명, 2. 게임시작, 3. 게임종료

게임설명 실행 후 다시 메뉴 프린트 화면으로 넘어가며(반복문 사용) 게임 시작 선택 시 game\_start 함수를 호출한다.

1. 게임을 시작하며 주사위를 굴림

* 주사위를 굴릴 경우 1~6까지의 숫자가 랜덤하게 선택되어 dice에 저장이 되고 dice 값을 map에 출력하기 위해 print\_map 함수를 불러온다. 이 dice 수는 계속해서 축적이 되고, 만약 11을 넘었을 경우 11을 빼서 0 부터 다시 축적하도록 한다.
* 1~11 숫자 중 일정한 칸에 도착하면 그 칸에 맞는 함수를 실행한다. Start, normal, mission, chance, rest 함수가 호출된다. 이것은 play\_juru 함수에서 switch 문을 통해 구현한다.

1. 함수 호출 시 기능

5개의 기능을 가진 칸에 알맞은 기능이 있다. 예를 들어 start 칸에는 drink 값을 그대로 유지하고 drink\_comp 값은 +1 해주어야 한다. 이때 drink값은 game\_start 함수에서 포인터 변수로 가져오는 것이 drink 값을 계산하기 가장 좋은 방법이다. Drink 와 drink\_comp의 값을 \*pdrink와 \*pcomp\_drink로 받아오고 각 함수에서 요구하는 사항만큼 값을 늘리고 줄인 뒤 game\_start 함수에 반환한다.

1. 미션 함수

미션 칸인 3과 9에 도착할 경우 369 게임과 베스킨라빈스 게임을 랜덤하게 선택하여 함수를 호출한다. 이때 rand()%2 코드를 사용하여 0이 나왔을때는 베스킨라빈스, 1이 나오면 369 함수를 불러온다.

1. 미션 게임 실행

* 369 게임의 경우 user와 comp중 랜덤하게 시작되며 remainder=num%10 으로 구성하여 remainder가 0, 3, 6, 9 가 되면 숫자 대신 0을 출력한다. User 가 틀리지 않고 21까지 출력하면 user가 성공한다.
* 베스킨라빈스 게임도 마찬가지로 선 공격이 user와 comp중 랜덤하게 시작되며 user가 1,2,3 을 입력하고 comp는 1,2,3 이 랜덤하게 배정된다. 31을 말하는 사람이 패하게 된다.

1. **프로그램 실행방법 및 예제**

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 텍스트이(가) 표시된 사진

   자동 생성된 설명텍스트, 점수판, 스크린샷이(가) 표시된 사진

   자동 생성된 설명 처음 실행 시 뜨는 화면이다. 1, 2, 3중 하나의 값을 입력받고 각 경우에 맞는 함수를 불러온다. 만약 다른 값을 입력하면 입력이 잘못되었다는 메시지가 뜬다.
2. 게임시작으로 넘어갔을 때 뜨는 화면, 말이 시작 위치에 놓인 것을 알 수 있고, USER와 COMP의 주량이 모두 0으로 초기화 되어 있는 것을 알 수 있다. 0을 누르면 play\_juru 함수를 호출하고 -1을 누를경우 화면 1로 돌아간다. 1,2,3,4,5,6을 입력하면 입력된 값만큼 말이 이동한다.
3. 주사위 굴린 후 ROUND 1 시작한다. 주사위 눈에 맞게 ‘O’ 말을 이동시키고 칸에 맞게 기능을 실행한다. 화면 3에서는 주사위 눈이 1 이므로 일반 칸(1)로 말이 이동하고, USER 1잔, COMPUTER 1잔 마신다.

한 잔씩 마시면 print\_drink 함수를 호출하여 잔 수가 1씩 증가한 것을 알 수 있다.

1. 텍스트이(가) 표시된 사진

   자동 생성된 설명다음 칸인 찬스 칸에 도착했을떄 화면이다. 찬스 칸에서 여명 808 과 세잔 더 중 한가지가 선택된다. 이 경우 여명 808이 선택되어 user의 주량이 0이 된 것을 볼 수 있다. 만약 현재 주량의 -2가 되어야 한다면 -1 주량이 되어 있지만, 음수 주량은 고려하지 않기 때문에 0으로 돌아온 것을 확인 할 수 있다

텍스트, 클래퍼보드이(가) 표시된 사진

자동 생성된 설명

1. 4라운드에서는 미션 칸에 도착하였다. 배스킨라빈스와 369 중 랜덤으로 369 게임이 실행된 것을 알 수 있다. 369 실행 시 선 공격자가 user로 랜덤하게 배정된 것도 확인할 수 있다. User가 처음에 1이 아닌 값을 입력할 시 comp의 승리가 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 369 게임 실행 시 21까지 성공적으로 말한 user가 승리한 것이 출력되고 컴퓨터가 4잔 마시게 되었다. Print\_drink 함수를 통해 누적 잔 수를 확인할 수 있다.
2. 텍스트이(가) 표시된 사진

   자동 생성된 설명10라운드에서 게임이 종료된 것을 확인할 수 있다. User가 패하며 게임 종료 시 comp의 주량이 늘지 않게 구현했으며 누가 이겼는지 마지막 줄에 말해준다.
3. **토론**

* While(1)을 사용해서 반복해야 하는 경우가 많았다. 함수안에서 동일 함수를 호출하는 재귀함수를 사용하였다. 문제를 일으킨 적은 없었지만 6주차 수업에 재귀함수의 사용이 메모리를 차지해 에러를 일으킬 수 있다는 것을 배워 재귀함수의 사용을 없앴다.
* 369 함수에서 3,6,9,0을 가진 수의 경우 0을 출력하도록 구성하는데 어려움이 있었다. 그러던 중 remainder라는 변수를 만들어 3,6,9,0을 나머지로 갖는 수를 모두 0으로 출력해내면 코드를 구성하는 것이 쉽겠다는 생각이 들어서 나머지 개념을 통해 함수를 구성했다.
* 또한 369 함수에서 입력 받는 num 값 외에 num1이라는 기준을 가진 변수를 만들어서 num1은 매 턴마다 수가 1 씩 증가하도록 설정했다. 입력 받은 num 값과 num1 값이 일치하기 않을 경우 무조건 computer 가 승리하도록 구성했고, 0을 예외로 지정하였다.

1. **결론**

* 태어나서 프로그램을 처음 짜본 것 이였기에 시행착오가 매우 많았다. 그럴때마다 lecture를 다시 재생하고 note를 반복해서 읽거나 assn-supp.c 파일을 만들어서 간단한 프로그램을 만들고 파일에 넣었다.
* 함수를 호출하고 함숫값을 받아오거나 포인터를 사용하는 일이 많았다. 함수를 호출하고 return 하는 원리를 마스터 한 것 같다.
* 또한 반복문(while, for)를 모두 사용하고 조건문 (if & switch)도 매우 많이 사용했다. 구문을 사용할 때 항상 머리로 먼저 컴파일 해보고 문제를 예측하고 수정하고 문제가 없다고 생각되었을 때 컴파일을 했다. 이 방식으로 진행하다 보니 반복문과 조건문이 어떻게 구성되는지 이해했다.

1. **개선 방향**

* 처음 코드를 짰을 때 반복되는 구문이 매우 많았다. 반복을 줄이기 위해 코트 수정을 많이 해야 했고, 특정 함수내에서 똑같은 함수를 불러내는 재귀 함수의 사용이 있어서 그 부분을 해결하기 위해 노력했다.
* 짧은 코드가 무조건 좋은 것은 아니지만 700자 정도가 나온 부분이 아쉽다. 최대한 효율을 높여 보기 위해 코드 수정을 반복했지만 더 이상 줄이기 어려웠다.
* 포인터 변수의 사용이 있었는데, 수업시간에 swap 함수로 잠시 다룬 내용이 전부여서 코드를 짜는데 어려움이 있었다. 향후 포인터 변수 심화 내용을 배우면 프로그램을 더욱 간결하게 구성할 수 있을 것이라고 생각한다.