과제 소스코드 설명

2017029407 김상은

1. 알고리즘

① 그리드 그리기

: 이중 리스트를 사용하여 이차원 그리드 형성, Greed[0][0]~Greed[5][7] 만들고 각각에 맞게 (#), (.), (@), (G), (S) 대입

② 가능한 이동 방향

: 4가지 경우 확인, 현재(S)의 위치를 기준으로 [+1][](S), [-1][](N), [][+1](E), [][-1](W) 내부의 기호를 살핀 후 (#)인 경우만 제외하고 가능한 경우로 생각, 방향 리스트를 만들어서 방향 검사한 후 각각 되는 경우의 알파벳을 N, E, S, W 순서대로 append 함수를 이용해 추가, 방향 출력한 후에는 방향 리스트를 다시 초기화시킨 후 다음 차례에 검사

: 이동 방향 입력 받은 후 방향 리스트에 in 함수 사용해서 false가 나오면 에러 메시지 출력, 다시 이동 방향 입력 받기, while문 써서 false 나오는 동안 계속 검사하게 함, 그러면 그동안에는 이동 카운트는 증가하지 않음

③ 이동 카운트

: for moveCount in range 21

: 중간에 게임을 이길 시 break문 걸어서 빠져나옴

: moveCount==21이면 게임을 진 것이므로 You Lose 띄우고 break문 걸어서 빠져나옴

④ 체크 포인트 카운트

: 이동 입력 받은 후 S가 이동한 위치의 기호가 (@)인 경우 체크포인트 +1

: 기호(@)를 기호 S로 바꾸고 전체 그리드 출력, 체크포인트 변수를 이용해 몇 번째 체크포인트인지 출력, 그리드 출력 후에 기호 S였던 위치를 (.)로 바꿈

2. 소스코드 간결화를 위한 노력

① 처음에는 현재 위치를 나타내는 변수와 이동할 위치를 나타내는 변수를 모두 사용하려고 했으나, 곰곰이 생각해 본 결과 위치를 나타내는 변수는 하나만으로도 충분한 것을 깨달았다. 그래서 불필요한 변수의 사용을 줄여 메모리 사용을 줄이고, 코드를 간단히 하였다.

② 체크 포인트가 몇 번째인지 출력하기 위해 order라는 리스트를 사용하였다. 체크 포인트에 도달할 때마다 체크 포인트 카운트 수가 증가하는데, 이 때 일일이 if-elif문으로 몇 번째인지 문장을 출력하는 것보다 서수를 나타내는 리스트를 하나 만들어서 문자열 간의 덧셈 연산으로 나타내는 것이 훨씬 간결하고, 불필요한 코드 수를 줄일 수 있을 듯 했다.

③ 가능한 한 함수를 정의해서 전역 변수의 수를 줄이고, 실행되는 부분을 간단히 나타내고자 했다. 한 문장을 출력하는 것은 함수를 정의해서 사용하기 보다는 본 함수에서 바로 나타내는 것이 더 나을 것 같아 따로 함수를 만들지 않았고, 미로를 출력하는 기능과 이동 가능 방향을 체크하는 기능, 이동 방향을 받고 이동가능한지 판단하는 기능 등 여러 줄로 구성된 것은 함수를 정의하여 사용했다.