<MAZE GAME>

팀명 : Pks(김주은, 이혜민)

작성일 : 2012.09.03

제출자 : 김주은(21000172),이혜민( 21100571)

과목명 : 프로그래밍 실습(월 목 3교시 분반)

과제내용 : 두개의 스택을 사용하여 MAZE 게임 만들기

1. 문제해석 :

미로 안에 있는 쥐를 작성하여라. 이 프로그램은 반드시 시작점에서 목적지까지의 지나왔던 길과 뒤로 물러났던 길을 포함하여서 출력하여야 한다. 따라서 만약 한 지점을 두 번 지나갔다면 이건 반드시 두 번 출력되어야 한다. 만약에 세 번 지나갔다면 세 번 출력되어야 한다.

이 미로는 다음 그림과 같이 보여진다. 쥐가 여행을 시작하는 입구는 이 프로그램을 수행하는 사용자에 의해 지정되어야 한다. 이것은 매번 바뀔 수 있다.

2차원 배열이 이 미로를 저장하는 데이터구조로 사용될 수 있다. 각각의 배열의 원소는 검정이거나 흰색일 수 있다. 검은 원소는 쥐가 들어갈 수 없는 곳이다. 검은 원소는 쥐가 이용할 수 있는 길이다. 배열의 검은 원소는 1로 표현될 수 있고 흰색원소는 0으로 표현된다.

쥐가 길을 지나갈 때 원소를 차례대로 지나간다. 즉, 쥐는 미로를 원소의 배열로 간주할 수 없다. 각각의 여행의 순간에서 단지 하나의 원소로 인식된다. (쥐는 path가 아닌 하나하나의 점으로 이동한다는 뜻). 이 원소를 currentSpot이라고 부르자.. 이것은 2개의 정수영역으로 구성된 structure로 표현될 수 있다. 처음 영역은 행이고 둘째 영역은 열이다. 이 두 가지로 좌표를 표현한다. 예를 들어 3-25의 5,12는 5행 12열의 원소이다.

이 문제는 미로를 생성하는 데서 시작된다. 그런 뒤 출구를 초기화하고 유저가 시작지점을 결정한다. 이 프로그램은 반드시 인내가 있어야 한다. 만약에 사용자가 검은 지점을 선택했다면 프로그램은 흰색지점이 선택될 때까지 반드시 다시 요청해야 한다. 쥐는 시작지점에서 출발해서 출구까지 도착하기 위해 노력한다. 하지만 어떤 시작지점은 출구로 도달할 수 없음을 명심해라.

쥐가 여행을 진행함에 따라, 경로를 찍는다. 한 지점에 들아 가면, 프로그램은은 그 지점의 클래스를 결정한다.

한 지점의 클래스는 다음 중에 하나와 같게 된다.

continuing ; 오직 이웃이 하얀색 지점이면 이 지점은 continuing spot이 된다. 즉, 쥐는 오직 하나의 선택밖에 할 수 없다.

intersection- 두 개 이상의 이웃이 하얀색이면 intersection spot이다. 즉, 쥐는 두 개 이상을 선택할 수 있다.

dead end- 이웃에 하얀색 지점이 없으면 dead end가 된다. 즉 쥐는 선택할 수 없고 되돌아간다.

d. exit- 쥐가 미로 밖으로 나갈 수 있으면 이 지점은 출구가 된다. 쥐가 출구를 찾으면, 쥐는 자유가 되고 치즈를 보상으로 받는다.

문제를 해결하기 위해 두 개의 스택이 필요하다. 첫 번째 스택은 쥐가 따라온 경로를 모두 출력하는 visit stack이다. 쥐가 한 지점을 도착할 때 마다 출구가 맞는지 제일 먼저 확인한다. 만약 아니면, 이 위치는 스택에 넣는다. 이 스택은 쥐가 dead end에 도달하고 반드시 돌아가야 할 때 사용 되어진다. 쥐가 지난 교차점으로 돌아갈 때 마다 backtrack 경로를 출력한다.

쥐가 교차점에 들어가면, 대안은 두번 째 스택에 위치한다. 이 교차점은 또한 특별한 decision token에 의해 표시되어야 하는데 이는 visitied stack에 위치한다.

decision token은 (-1,-1)로 조정된다. 경로를 선택하기 위해 대안은 alternative 스택으로부터 pop되고 쥐는 계속해서 경로를 지나간다. 되돌아가는 동안, 만약 쥐가 decision token에 도달하면, 토큰은 버려지고 다음 대안이 alternative스택으로부터 선택된다. 이 지점은 옆의 위치에 에스터리스크(\*)로 출력되고 다음 alternative 경로가 선택되어짐을 보여주기 위한 것이다.

문제해결과정

1. Maze를 나타내는 data type을 2차원 array로 결정하였다.
2. 좌표를 나타내기 위한 structure 를 설정하였다.
3. 지나온 경로를 나타내는 trace stack과 교차점을 나타내는 alternative stack을 structure array 로 만들었다.
4. Move하는 함수와 move하기 위한 조건 checking 알고리즘을 만들었다..
5. Move 함수에서 이동방향의 우선순위를 설정하였다.
6. Dead end에서 back tracking 을 위한 algorithm과 함수를 작성하였다.
7. 모든 sub함수를 먼저 구현한 후 main 함수를 작성하였다.

구현방법

1. 좌표를 나타내는 structure를 define한 후 현재 지점을 나타내는 currentSpot과 출구지점을 나타내는 exitSpot을 정의하였다.
2. 2차원 array를 통하여 maze를 구현하였다.
3. Maze를 콘솔 창에 출력시킨 후, 사용자로부터 미로게임의 시작점을 입력 받는다.
4. Maze 크기에 알맞은 값이 입력되면, 미로게임이 시작되고, 올바른 값이 입력될 때까지 while문으로 반복해서 입력받는다.
5. 이동방향을 동서남북으로 지정하고, 우선순위를 정하여 움직이도록 구현하였다. 🡪if 조건문에 현재 위치값과 다음 위치값을 0인지 비교하여 0이면 이동하도록 하였다.
6. 이동할 때마다, 모든 경로는 trace stack에 push하여 넣고 한번 지나간 지점의 값은 지나가지 않은 지점과 비교하기 위해 ‘2’값을 넣었다. 또한 이동 할 때마다 print Spot을 하여 경로를 알 수 있게 하였다.
7. 교차점에 도달했을 때는 카운트 변수를 사용하여 현재위치로부터 이웃 지점의 0이 한 개 이상이면 alternative stack에 push하도록 하였다.
8. 막다른 길(Dead-end)에 도달했으면, trace stack의 값을 pop하여 가장 가까운 교차지점까지 backtracking 한다.
9. 교차 지점에서 가지 않았던 다른 길로 이동한다.
10. 만약에 교차점에서 갈 수 있는 길을 모두 다 갔다면 alternative stack을 pop하여 이전 교차지점으로 backtracking 한다.
11. 만약 모든 지점을 돌아서 원래 시작지점으로 돌아오면 (더 이상 backtracking 할 수 없고 trace stack이 empty상태이면) 출구가 없는 것으로 판단하고 종료한다.
12. 만약 currentSpot과 exitSpot의 row와 column 이 일치하면 출구를 찾은것으로 판단하고 종료한다.

시행착오

1. Move를 위한 조건문에서 current Spot을 maze의 col, row로 지정하지 않았다.

Ex )maze[currentSpot.row][currentSpot.col+1]==0 이렇게 해야 하는데

currentSpot.col+1 == 0 이렇게 해서 move가 정상적으로 작동하지 않았다.

1. back track을 요청했을 때 pop을 두 번 하는 경우가 생기게 되어서 traverse가 끝나지 않았는데도 stack이 empty가 되는 실수가 있었다.
2. Maze의 범위를 벗어나는 곳으로 가는 조건을 check하지 않아서 current spot이 maze바깥으로 나가버리는 실수가 있었다.
3. Exit check를 처음 시작할 때 하지 않아서 만약 exit 지점이 처음 시작 지점이라면 exit를 찾지 못하는 경우가 발생하였다.
4. north move함수의 방향을 잘못 지정하여서 current spot이 이상한 점으로 이동되는 경우가 있었다.

평가

1. 서브프로그램을 이용하여 메인 함수의 구현을 간단하게 하였다. 🡪프로그래머의 가독성을 높였다.
2. Mouse의 이동방향을 좌표와 함께 표시하여서 어떻게 이동했는지 알기 쉽게 하였다.
3. 모든 이동 가능한 지점을 traverse함으로 exit 지점이 존재한다면 확실하게 찾을 수 있게 하였다.
4. 문제의 조건에서 decision token과 asterisk에 관련된 부분을 제외하고 maze의 모든 조건을 만족시켰다.
5. Maze가 매번 바뀌지 않고 프로그래머가 수동으로 직접 입력해야 하는 번거로움이 있다.

<실행결과>

