자료구조 Data Structure | 조행래

스택과 큐

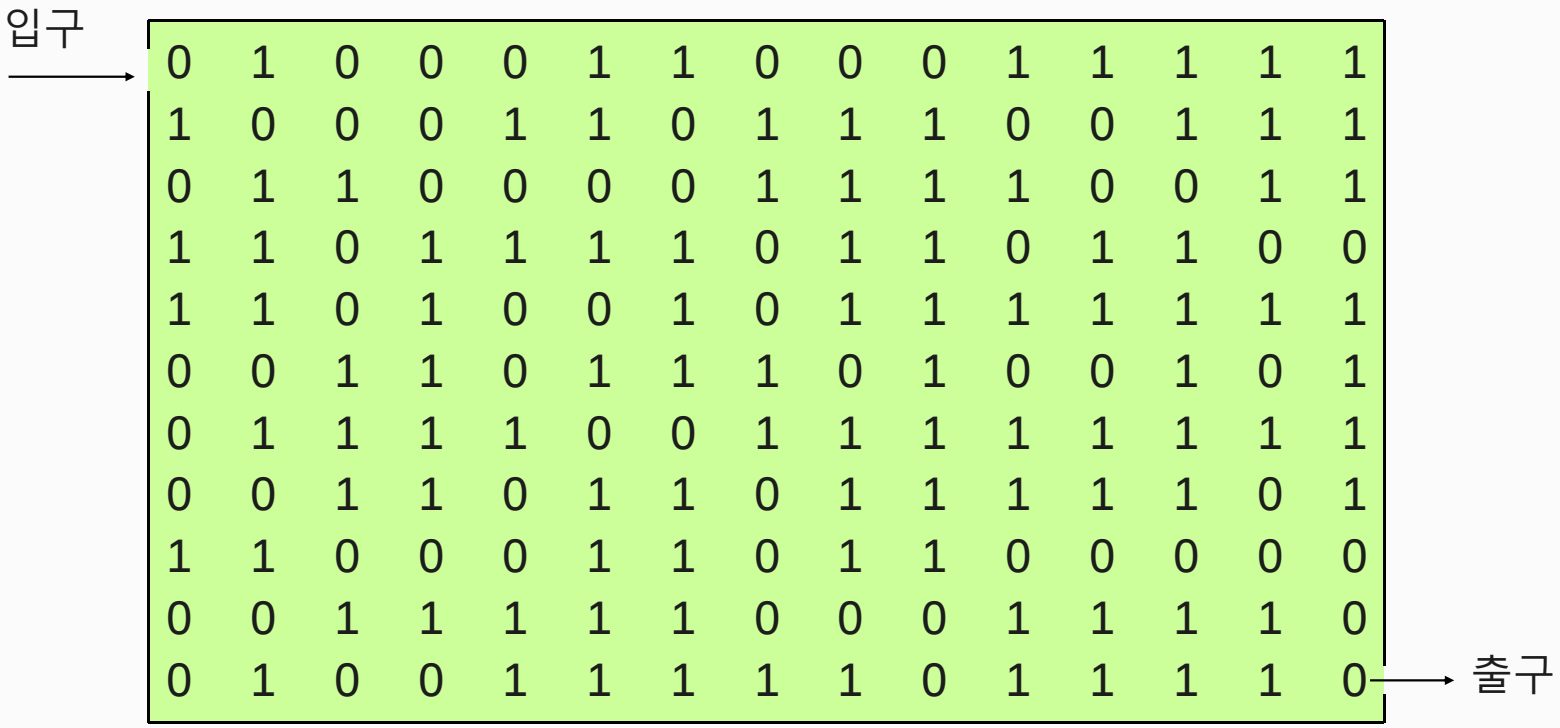
스택과 큐의 응용(1): 미로 찾기

학습 목표

- 배열을 이용하여 미로 찾기 문제를 표현한다.
- C-언어에서 스택을 이용하여 미로 찾기 문제의 해결 알고리즘을 구현할 수 있다.

1. 문제 소개

- **미로 찾기 문제(Mazing Problem)**
  - 입력: 길과 벽으로 구성된 미로, 입구와 출구
  - 출력: 입구에서 출구까지의 경로

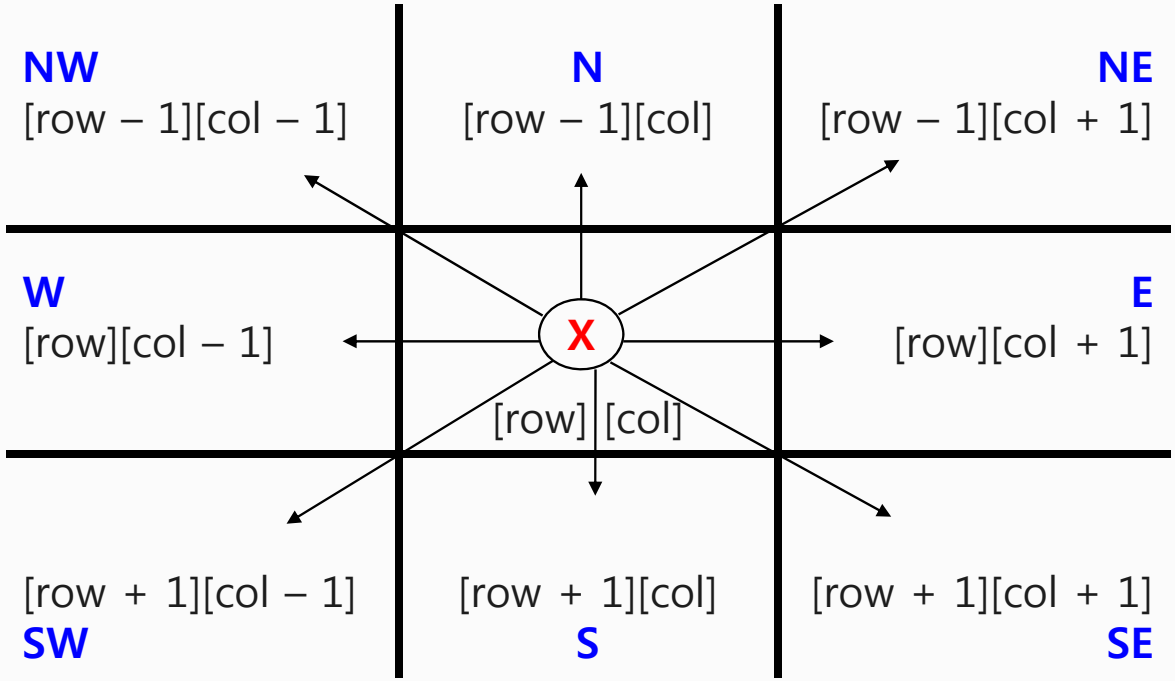


2. 문제의 표현

- **2차원 배열을 이용하여 미로를 구현**
  - **maze[row][column]**
  - 0 – 길, 1 – 벽
  - 이동 경로: (row, column) 좌표들의 순서 리스트
    - 시작점: 입구의 좌표
    - 종점: 출구의 좌표

- **현재 위치에서 값이 0인 다음 위치로 이동 가능**
  - 8방향으로 이동 가능(N, NE, E, SE, S, SW, W, NW)
  - 주의: 경계 지역으로 8방향이 아님
    - 모서리: 3방향, 변: 5방향
  - m × p 미로를 (m + 2) × (p + 2) 미로로 변환
  - 각 경계 영역의 maze 배열값은 1로 설정
  - 입구: maze[1][1], 출구: maze[m][p]

현재 위치에서 이동가능 지점



현재 위치에서 8개의 방향좌표를 계산하기 위하여
좌표들간의 차이를 구조체 배열로 정의

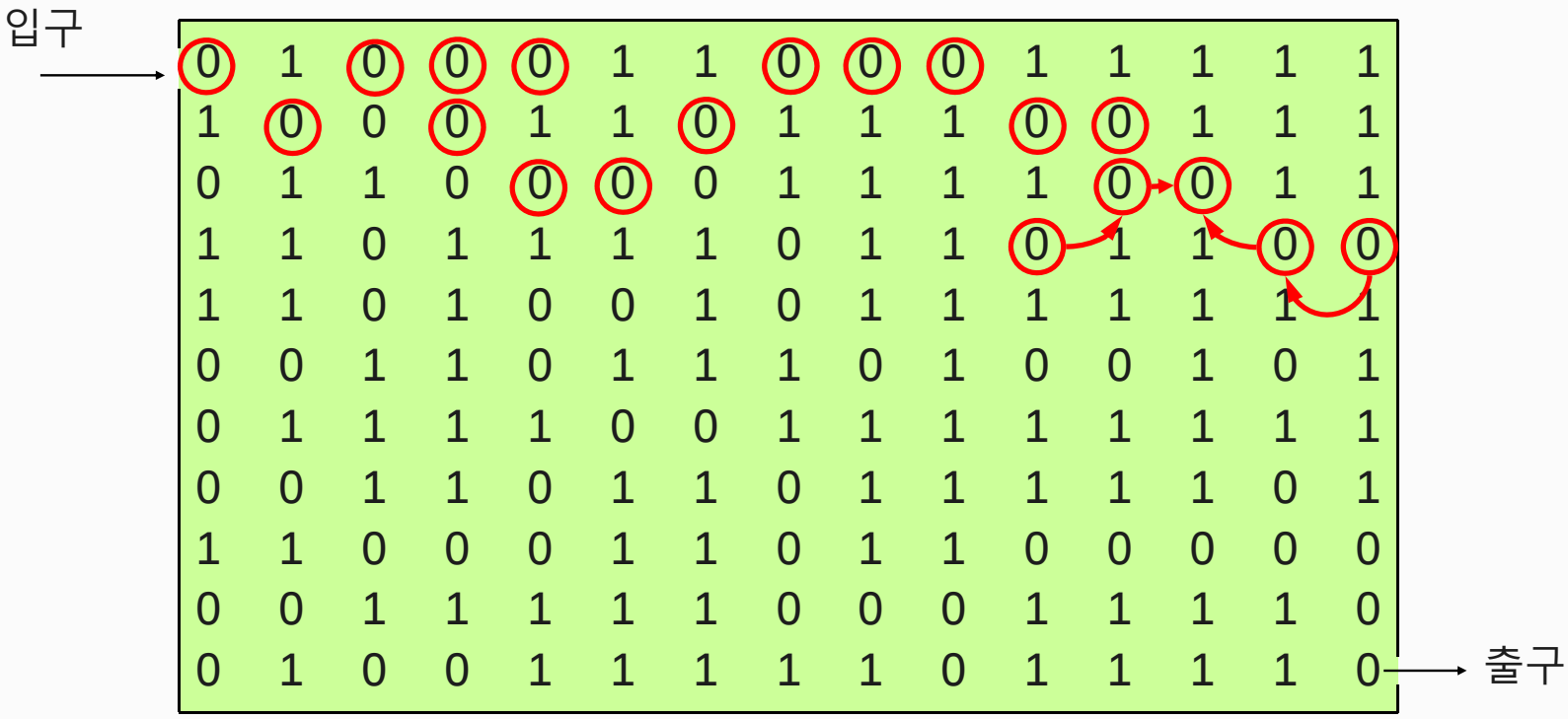
이동방향의 표현

- **8가지 이동방향을 표현하기 위해 move[8] 배열 사용**

```
typedef struct {
        short int       x;
        short int       y;
} offsets;
offsets move[8]; // move[0]이 북쪽 → 시계 방향
```

Name	Dir	move[dir].x	move[dir].y
N	0	-1	0
NE	1	-1	1
E	2	0	1
SE	3	1	1
S	4	1	0
SW	5	1	-1
W	6	0	-1
NW	7	-1	-1

3. 알고리즘의 동작 과정


입구
0 1 0 0 0 1 1 0 0 0 1 1 1 1 1
1 0 0 0 1 1 0 1 1 1 0 0 1 1 1
0 1 1 0 0 0 0 1 1 1 1 0 0 1 1
1 1 0 1 1 1 1 0 1 1 0 1 1 0 0
1 1 0 1 0 0 1 0 1 1 1 1 1 1 1
0 0 1 1 0 1 1 1 0 1 0 0 1 0 1
0 1 1 1 1 0 0 1 1 1 1 1 1 1 1
0 0 1 1 0 1 1 0 1 1 1 1 1 0 1
1 1 0 0 0 1 1 0 1 1 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0 1 1 1 1 0
0 1 0 0 1 1 1 1 1 0 1 1 1 1 0
출구

4. 구현 방안

- **현재 좌표가** (row, col)**일 경우, 다음 이동좌표의 계산**
  - next_row = row + move[dir].x
  - next_col = col + move[dir].y
- **북쪽**(N)**부터 다음 이동좌표를 차례대로 계산하여 이동 가능한 지**(즉, **길** or **벽**) **확인한 후, 길이면 이동.**
- **한번 갔었던 길을 다시 갈 필요는 없으므로, 기존에 다녀온 길들을 mark[m+2][p+2] 배열에 저장**
  - 모든 mark[row][col]의 값은 0으로 초기화
  - maze[i][j]를 방문한 후, mark[i][j]를 1로 설정
- **갔다가 길이 없을 경우 돌아와야지? stack[] 사용**

```
#define MAX_STACK_SIZE 100 // = m × p
typedef struct {
    short int    row;
    short int    col;
    short int    dir;
} element;
element stack[MAX_STACK_SIZE];
```

5. 알고리즘 구현 – 초기 버전

```
미로의 입구좌표와 N 방향으로 stack 초기화    //최적화?
while ( stack is not empty )    {
   // stack top의 위치로 이동
   <row, col, dir>  =  delete from top of stack;
   while ( there are more moves from current position ) {
         <next_row, next_col>  =  coordinate of next move;
         dir  =  direction of move;
         if ((next_row == EXIT_ROW) &&
              (next_col == EXIT_COL))
                        success;
```

알고리즘 구현 – 초기 버전(계속)

```
            if (maze[next_row][next_col] == 0 &&
                        mark[next_row][next_col] == 0)    {
                        // 정상적인 길이며 아직 방문한 적이 없음
                        mark[next_row][next_col] = 1;   // 이제 방문
                                                // 현재 좌표와 방향을 stack에 저장
                        add <row, col, dir> to the top of the stack;
                        row = next_row;
                        col = next_col;
                        dir = north;
            }
      }
}
printf( "No path found ₩n" );
```

6. 알고리즘 구현 – 최종 버전

```
void path(void)
{   // 미로를 통과하는 경로가 존재할 경우, 이를 출력
    int  i,  row,  col,  next_row,  next_col,  dir;
    int  found = FALSE;
    element  position;

    // 미로의 입구좌표와 E 방향으로 stack 초기화
    mark[1][1] = 1;  top = 0;
    stack[0].row = 1;  stack[0].col = 1;  stack[0].dir = 2;
    while ( top > -1  &&  !found )    { // stack이 empty가 아니고, 아직
                                        // 경로를 발견 못할 때까지 실행
        position = pop();          // top의 위치로 이동
        row = position.row;        col = position.col;
        dir = position.dir;
```

알고리즘 구현 – 최종 버전 (계속)

```
    while ( dir < 8  &&  !found )    { // 8방향을 차례대로 검사
        next_row = row + move[dir].x;     // move in direction dir
        next_col = col + move[dir].y;
        if ( next_row  ==  EXIT_ROW  &&  next_col  ==  EXIT_COL )
            found = TRUE; // 출구 발견. 경로는 어디에?
        else if ( !maze[next_row][next_col]  &&
            !mark[next_row][next_col] )    { // 아직 안 가본 길
                mark[next_row][next_col] = 1;
                position.row = row;
                position.col = col;
                position.dir = ++dir;
                push(position);      // 현재 좌표와 방향을 stack 저장
                row = next_row;    // 안 가본 길로 전진. 방향은 북쪽
                col = next_col;
                dir = 0;
        }
        else  ++dir;
}   }
```

알고리즘 구현 – 최종 버전 (계속)

```
    if (found)    {          // stack에 저장된 경로 출력
            printf( " The path is: ₩n " );
            printf ( "row  col ₩n" );
            for ( i=0;  i <= top;  i++ )
                    printf( " %2d %5d ",  stack[i].row, stack[i].col );
            printf( " %2d %5d ₩n ", row, col );
            printf( " %2d %5d ₩n ", EXIT_ROW, EXIT_COL );
    }
    else printf( " The maze does not have a path ₩n " );
}
```

요약 정리

- 배열을 이용하여 미로를 표현하는 방법을 설명
- 스택을 이용하여 미로 찾기 문제를 해결하는 알고리즘 설명