DFS 미로 탐색
(C++사용. 파일로부터 맵 입력받기)

- N x M 크기의 배열로 표현되는 미로가 있다.
- 미로에서 1은 이동할 수 있는 칸
  0은 이동할 수 없는 칸을 나타낸다.
- 왼쪽 위 (1, 1)에서 출발하여 오른쪽 아래 (N, M)의 위치로 이동한다
- 이 때 지나는 모든 좌표와 전체 칸 수를 출력하시오.
- 이동할 때, 서로 상하좌우로 인접한 칸으로만 이동할 수 있다.
- 칸을 셀 때에는 시작 위치와 도착 위치도 포함한다

[입력 파일]

4 6 ← 행의 수, 열의 수

1 0 1 1 1 1

1 0 1 0 1 0

1 0 1 0 1 1

1 1 1 0 1 1

"miro_input.txt"

[실행결과]

(0,0) (1,0) (2,0) (3,0)
(3,1) (3,2) (2,2) (1,2)
(0,2) (0,3) (0,4) (1,4)
(2,4) (2,5) (3,5) 총
15개

DFS 미로 탐색
(C++사용. 파일로부터 맵 입력받기)

◆ 사용하는 자료구조

- int* map = NULL; // 맵 읽어들이기.
- int* path = NULL; // 돌아다닌 경로 표시.
- deque<pair<int, int>> _stack; // 좌표 보관할 스택.
- int x = 0, y = 0; // 현재 위치.

◆ 수행 순서

- 시작좌표 (0,0)을 스택에 push_back
- while (!도착 and !스택 empty)
  - ❖ 스택 top 위치로부터 옮길 위치를 찾기 -> 스택에 push_back
  - ❖ 못 찾았으면 스택에서 pop_back
- 스택에서 모든 좌표를 꺼내 출력 (front()에서)하고 총 좌표 수도 출력.

DFS 미로 탐색 (1/5)
(C++사용. 파일로부터 맵 입력받기)

```
#include <cstdio>
#include <fstream>
#include <iostream>
#include <vector>
#include <queue>
#include <deque>

using namespace std;
```

DFS 미로 탐색 (2/5)
(C++사용. 파일로부터 맵 입력받기)

```
int main(void) {
        int rows, cols;
        ifstream rFile("miro_input.txt");
        int* map = NULL;        // 맵 읽어들이기.
        int* path = NULL;       // 돌아다닌 경로 표시.
        deque<pair<int, int>> _stack;// 좌표 보관할 스택.
        int x = 0, y = 0;       // 현재 위치.
        if (rFile.is_open())        {
                rFile >> rows >> cols;
                map = new int[rows * cols];
                path = new int[rows * cols];
                int i = 0;
                for (int row = 0; row < rows; row++) {
                        for (int col = 0; col < cols; col++) {
                                rFile >> map[i++];
                                path[row * cols + col] = 0;
                        }
                }
        }
        .....
```

DFS 미로 탐색 (3/5)
(C++사용. 파일로부터 맵 입력받기)

```
// 시작 지점을 push하고 탐색 시작.
_stack.push_back(make_pair(y, x));
path[y * cols + x] = 1;
bool arrived = false;
while (!_stack.empty() && !arrived) {
        y = _stack.back().first;
        x = _stack.back().second;
        bool moved = true;
        while (moved) {
                // 목표 지점인가?
                if (y == rows - 1 && x == cols - 1) {
                        arrived = true;
                        break;
                }
                .....
        }
}
```

현재 위치가 stack top에 있으니 그 값을 기준으로 움직일 자리 찾기.

DFS 미로 탐색 (4/5)
(C++사용. 파일로부터 맵 입력 받기)

```
moved = false;
if (x > 0 && map[y * cols + (x - 1)] == 1
    && path[y * cols + (x - 1)] == 0) {
    x = x - 1; moved = true; }
else if (y > 0 && map[(y - 1) * cols + x] == 1
    && path[(y - 1) * cols + x] == 0) {
    y = y - 1; moved = true; }
else if (x < cols - 1 && map[y * cols + (x + 1)] == 1
    && path[y * cols + (x + 1)] == 0) {
    x = x + 1; moved = true; }
else if (y < rows - 1 && map[(y + 1) * cols + x] == 1
    && path[(y + 1) * cols + x] == 0) {
    y = y + 1; moved = true; }
```

left, top, right, bottom 순으체크.
- 경계값 체크
- 길인지(1인지) 체크
- 방문 안했는지 체크.

```
if (moved) {
    path[y * cols + x] = 1;
    _stack.push_back(make_pair(y, x));
}
```

옮겨갈 자리 찾았으면 stack과 path에 방문표시.

```
else
    _stack.pop_back();
```

옮겨갈 자리 못 찾았으면 stack 에서 하나 꺼냄.

DFS 미로 탐색 (5/5)
(C++사용. 파일로부터 맵 입력받기)

```
    // 스택의 모든 좌표를 출력.
    int count = 0;
    while (!_stack.empty()) {
        cout << "(" << _stack.front().first << ","
            << _stack.front().second << ")" << " ";
        _stack.pop_front();
        count++;
    }
    cout << "총 " << count << "개";
    return 0;
}
```

[실행결과]
(0,0) (1,0) (2,0) (3,0) (3,1) (3,2)
(2,2) (1,2) (0,2) (0,3) (0,4) (1,4)
(2,4) (2,5) (3,5) 총 15개

BFS 미로 탐색
(C++사용. 표준입력에서 맵 입력받기)

- N x M 크기의 배열로 표현되는 미로가 있다.
- 미로에서 1은 이동할 수 있는 칸
  0은 이동할 수 없는 칸을 나타낸다.
- 왼쪽 위 (1, 1)에서 출발하여 오른쪽 아래 (N, M)의 위치로 이동한다
- 이동할 때 지나야 하는 최소의 칸 수를 구하시오.
- 이동할 때, 서로 상하좌우로 인접한 칸으로만 이동할 수 있다.
- 칸을 셀 때에는 시작 위치와 도착 위치도 포함한다

[입력 파일]

4 6 ← 행의 수, 열의 수
101111
101010
101011
111011

"miro_input.txt"

[실행결과]

15

BFS 미로 탐색
(C++사용. 표준입력에서 맵 입력받기)

◆ 사용하는 자료구조

- struct TimePos { // 위치와 시간을 묶음.
  int y; int x; int time;
  };
- string arr[102]; // 맵 읽어들이기
- bool* visit = NULL; // 동적배열. 방문여부표시.
- queue<TimePos> q; // 큐
- int dy[4] = { -1,0,1,0 }; // 방향 설정
- int dx[4] = { 0,-1,0,1 };

◆ 수행 순서

- 시작좌표 (0,0)을 큐에 push+방문표시
- while (!큐 empty)
  - ❖ 큐에서 x, y, cnt 를 꺼냄
  - ❖ 목표지점이면 cnt를 출력하고 break
  - ❖ 4방향 좌표 중 조건에 부합되면 큐에 push+방문표시.

실습
bfs.cpp

BFS 미로 탐색
(자료구조와 알고리즘)

[설명용 데이터]
입력
3 3
111
110
011
출력
5

```
#include <iostream>
#include <queue>
using namespace std;
struct TimePos {
        int y;          // row
        int x;          // col
        int time;
};
int main() {
        string arr[102];
        bool *visit = NULL;

        // 방향 설정
        int dy[4] = { -1,0,1,0 };
        int dx[4] = { 0,-1,0,1 };
        int rows, cols;
```

큐에 보관할 내용

맵 정보 읽어 들일 자리.
숫자라도 1개짜리가 연이어 들어올 때는 string이용이 편리.

맵 크기에 따라 동적으로 할당 받을 예정.

4방향 좌표 구할 때 사용.

맵의 크기.

실습
bfs.cpp

BFS 미로 탐색

```
int main() {
        ……
        cin >> rows >> cols;

        visit = new bool[rows*cols];
        memset(visit, 0x00, sizeof(bool) * rows * cols);
        for (int i = 0; i < rows; i++)
                cin >> arr[i];

        queue<pair<XY, int>> q;
        q.push({ { 0,0 }, 1 });
        visit[0] = true;

        ……
        return 0;
}
```

입력 받은 col, row 개수에 따라 메모리 할당.
할당받은 메모리는 쓰레기값이므로 0으로 초기화.

큐 생성.
좌표 (y,x) 와 cnt 추가.
+ 방문표시.

실습
bfs.cpp

BFS 미로 탐색

```
while (!q.empty()) {
    int y = q.front().y;
    int x = q.front().x;
    int cnt = q.front().time;
    q.pop();
    if (y == rows - 1 && x == cols - 1) {
        cout << cnt;
        break;
    }
    for (int i = 0; i < 4; i++) {
        int nextY = dy[i] + y;
        int nextX = dx[i] + x;
        if (nextY < 0 || nextY >= n ||
            nextX < 0 || nextX >= m ||
            visit[nextY*cols+nextX] || !arr[nextY][nextX])
            continue;
        q.push({ {nextY, nextX}, cnt + 1 });
        visit[y * cols + x] = true;
    }
}
delete [] visit;
```

도착했다면 그 값 출력(출력하는 순간이 최소)

4방향 좌표에 대해서 조건에 부합되면 큐에 push + 방문표시.