1. 물이 흐르는 것을 표현하기 위해 구현한 알고리즘 및 자료구조

<ofApp.h>

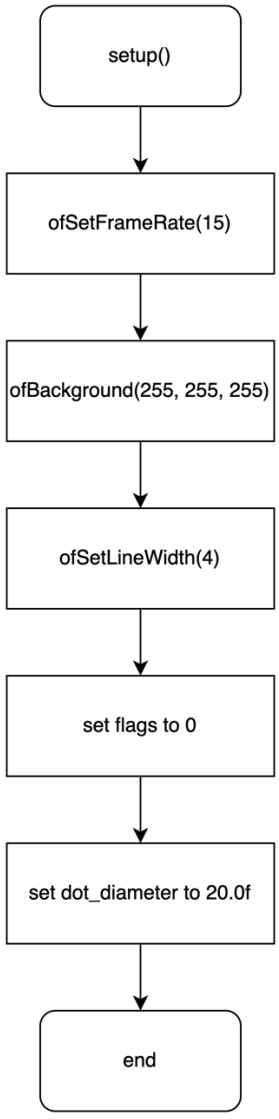
```
#pragma once

#include "ofMain.h"

class ofApp : public ofBaseApp{

    public:
        void setup();
        void update();
        void draw();

        void keyPressed(int key);
        void keyReleased(int key);
        void mouseMoved(int x, int y );
        void mouseDragged(int x, int y, int button);
        void mousePressed(int x, int y, int button);
        void mouseReleased(int x, int y, int button);
        void mouseEntered(int x, int y);
        void mouseExited(int x, int y);
        void windowResized(int w, int h);
        void dragEvent(ofDragInfo dragInfo);
        void gotMessage(ofMessage msg);

    /* WaterFall-related member variables Regions */

    // flag variables
    int draw_flag;
    int load_flag;
    int waterfall_flag;

    // Line segment and dot related variables
    int num_of_line, num_of_dot;
    float dot_diameter;

    /* ========== week1 ========== */
    vector<int> lx, ly, rx, ry;
    vector<int> dot_x, dot_y;
    int water_dot = 0;
    /* ========== week1 end ========== */

    /* ========== week2 ========== */
    vector<double> slope;
    vector<int> path_x, path_y;
    int path_idx = 0;
    /* ========== week2 end ========== */

    /* WaterFall-related member functions */

    void processOpenFileSelection(ofFileDialogResult openFileResult);
    void initializeWaterLines();

};
```

'l'키를 누르면 입력 파일로부터 선반의 위치와 점의 위치를 입력 받아서 vector container를 이용해 x좌표와 y좌표를 차례대로 저장했다. 이때 선반을 입력받으면서 기울기(slope)도 계산해서 저장했다. 그리고 'd'키를 누르면 선반과 점을 화면에 그려주었다. 물이 나오는 점은 빨간색으로,

물이 나오지 않는 점은 검은색으로 그려주었고, water_dot이라는 변수를 사용해 이 값(0~) 번째에 있는 점이 물이 나오는 점인 것을 알 수 있도록 했다.

- setup()

frame rate, background color, line width, 점의 지름 길이를 설정해주고 flag를 모두 리셋한다.

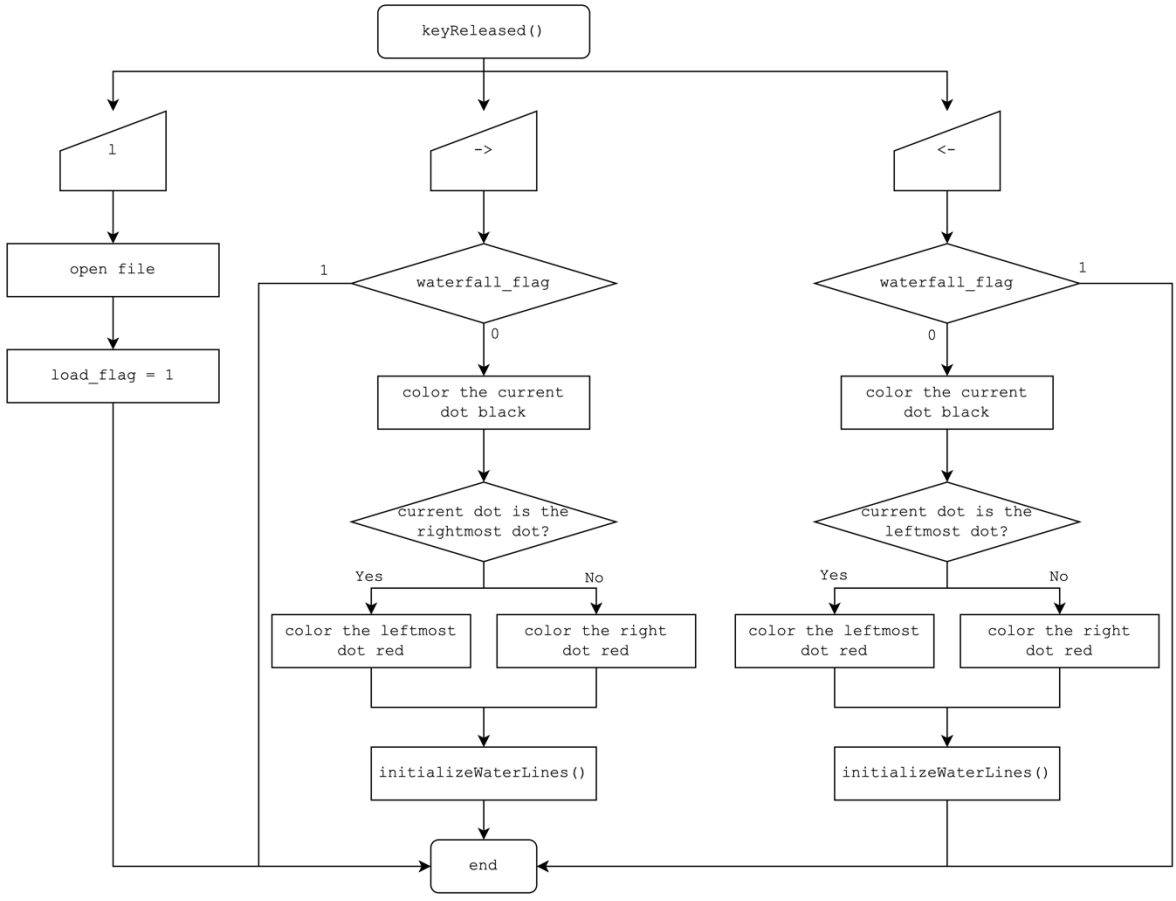


- keyReleased()

'l'을 누르면 파일을 열고 load_flag를 1로 set한다.

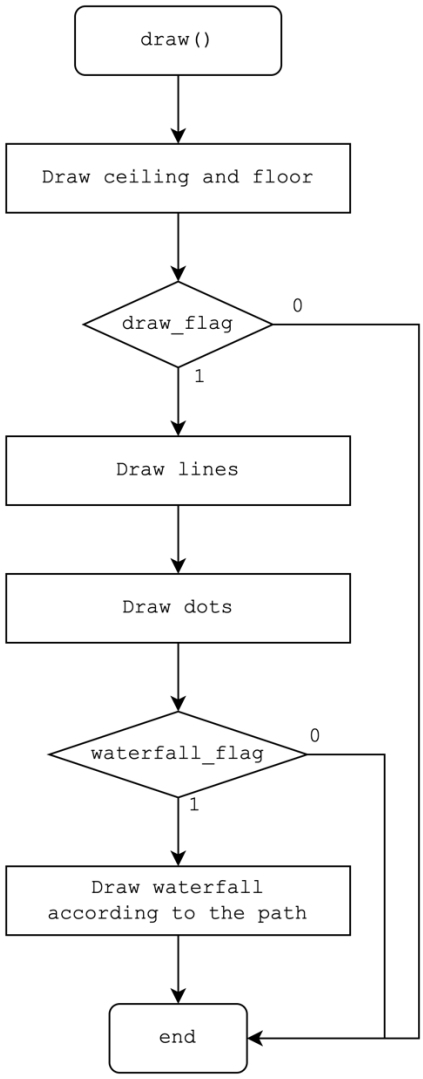
'→'를 누르면 물이 나오는 점(빨간색 점)을 현재 점의 오른쪽 점으로 변경한다. 만약 현재 점이 가장 오른쪽 점이었다면 가장 왼쪽 점으로 변경한다. 단, 물이 흘러나오고 있는 상태라면 물이 나오는 점의 위치가 변경되지 않는다. 위치를 변경했다면 그 위치에서 물이 흘러나온다면 어느 곳으로 갈 지 path를 구한다.

'←'를 누르면 물이 나오는 점(빨간색 점)을 현재 점의 왼쪽 점으로 변경한다. 만약 현재 점이 가장 왼쪽 점이었다면 가장 오른쪽 점으로 변경한다. 단, 물이 흘러나오고 있는 상태라면 물이 나오는 점의 위치가 변경되지 않는다. 위치를 변경했다면 그 위치에서 물이 흘러나온다면 어느 곳으로 갈 지 path를 구한다.



- draw()

천장과 바닥을 그린다. draw_flag가 1이라면 선반과 점을 그리고, water_flag가 1이라면 물줄기도 path에 따라 그린다.



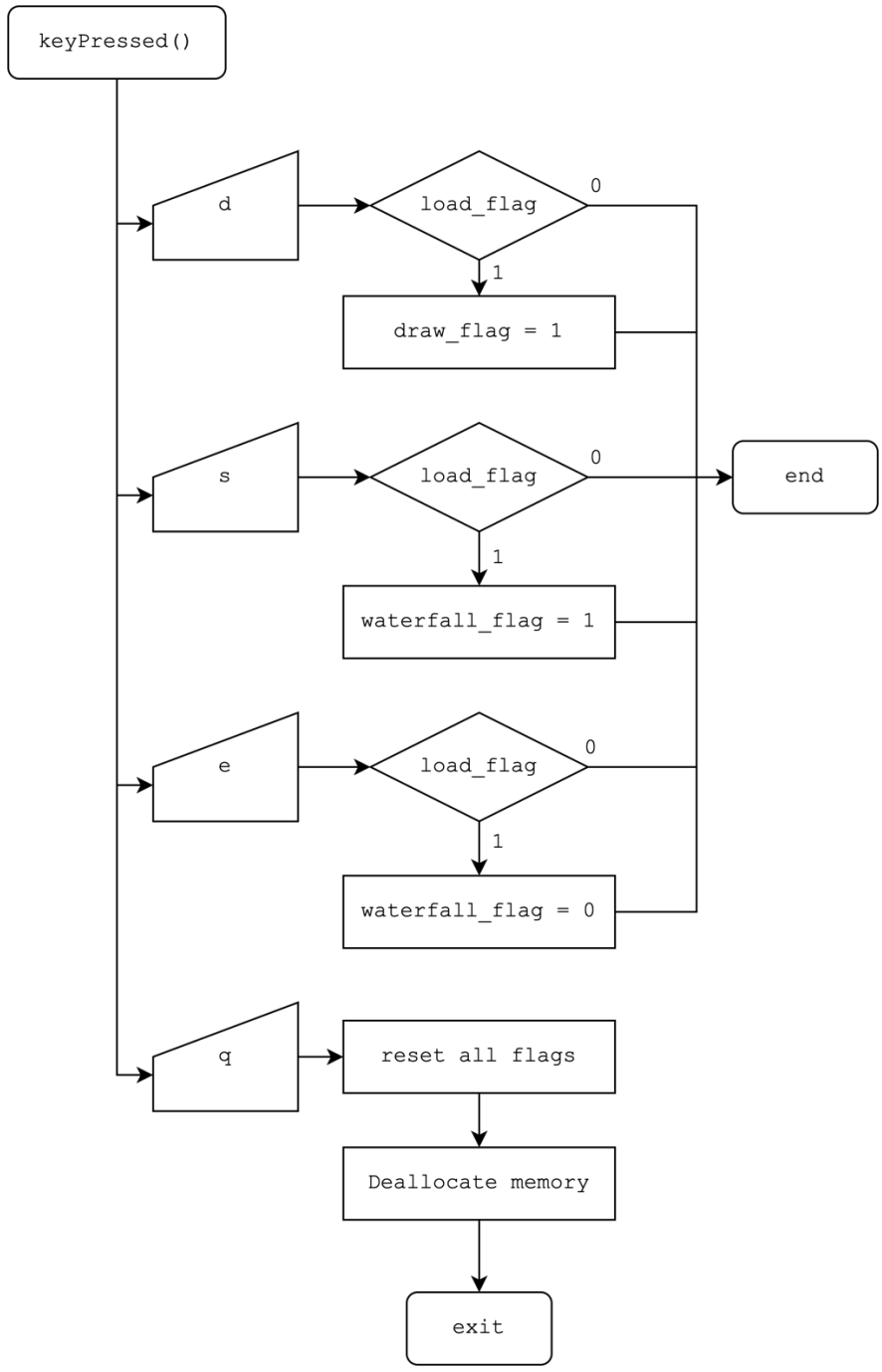
- keyPressed()

'd'를 누르면 load_flag가 1일 때 draw_flag을 1로 set한다.

's'를 누르면 load_flag가 1일 때 waterfall_flag를 1로 set한다.

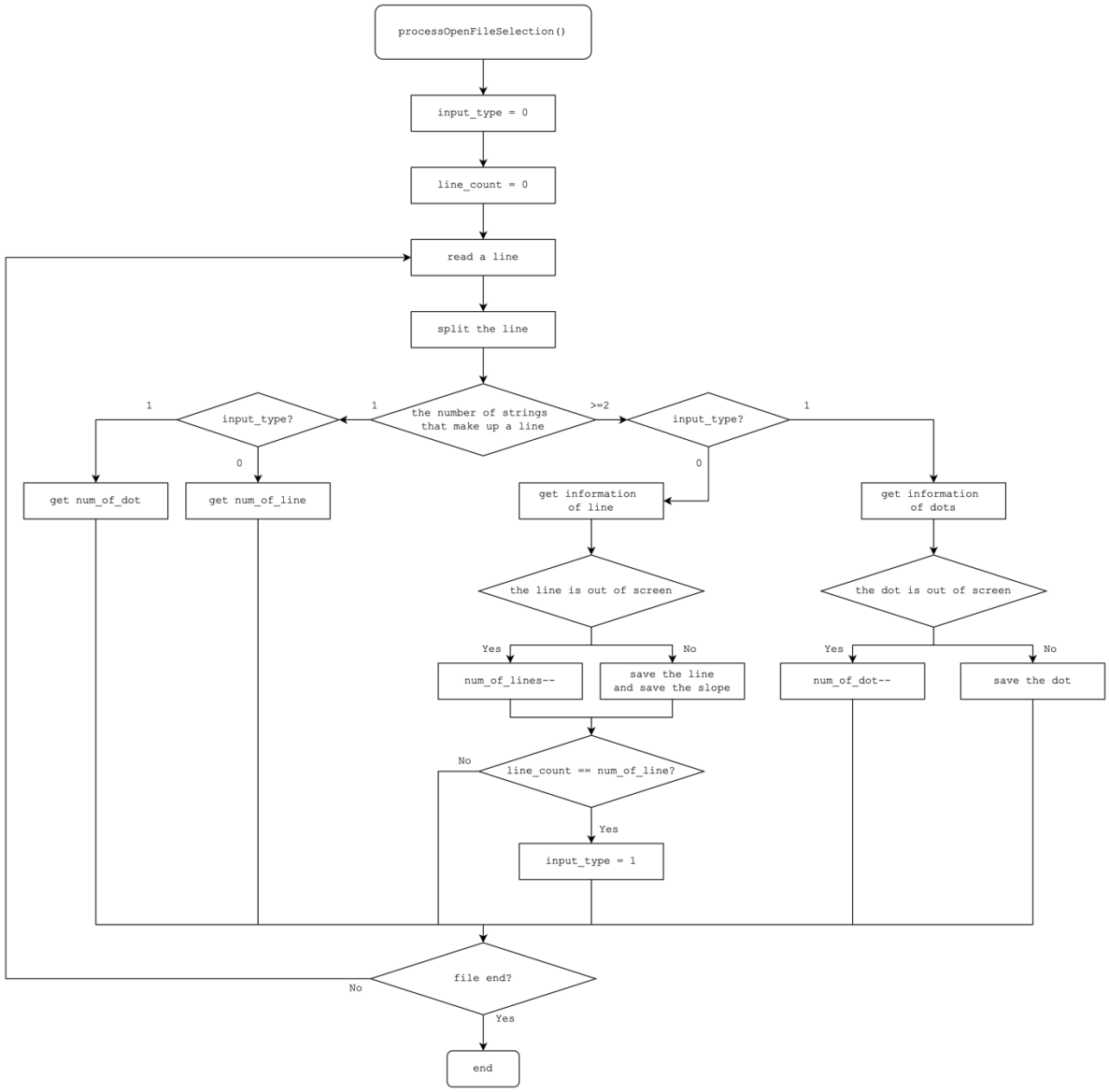
'e'를 누르면 load_flag가 1일 때 waterfall_flag를 0으로 reset한다.

'q'를 누르면 모든 flag를 reset하고, 동적으로 할당된 메모리를 모두 해제한 후, 프로그램을 종료한다.



- processOpenFileSelection()

input_type을 0으로 설정하고, 선반의 정보를 먼저 입력받고 선반의 정보를 모두 입력받은 뒤에는 input_type을 1로 설정해서 점의 정보를 입력받는다. 각 좌표는 화면을 벗어나지 않는지 체크한 뒤 vector를 이용해 저장했다. 예를 들어 선반의 정보가 a b c d라면 lx에 a, ly에 b, rx에 c, ry에 d를 차례대로 저장했다. 따라서 i번째 선반의 정보를 알고 싶다면 각 lx, ly, rx, ry에서 i번째 원소를 참고하면 된다. 또한 선반 정보를 저장할 때 (y2-y1)/(x2-x1)을 계산해 slope로 저장했다. slope은 double 타입의 vector를 이용해 저장했다.



- initializeWaterLines()

먼저 x, y를 현재 물이 나올 점의 좌표로 저장한다. 그리고 path에 저장한다. path는 path_x, path_y의 vector를 활용했다. 그리고 모든 선반에 대해 x가 선반의 왼쪽 x좌표와 선반의 오른쪽 x좌표 사이에 있다면 그 선반의 직선방정식과 물줄기가 만나는 지점의 y좌표를 구했다. 그리고 이 지점이 현재 지점보다 아래에 있다면 그 물줄기가 떨어져 만나는 지점을 path에 저장했다. 만약 선반에 물줄기가 떨어졌다면, 선반이 기울어진 방향을 따라 물이 흘러가야 하므로, 선반의 y좌표가 더 큰 방향으로 물줄기가 흘러갈 수 있게 그 선반의 끝 지점을 path에 저장했다. 그리고 최종적으로 구한 지점을 x, y로 set하고, 위와 같은 과정을 물이 바닥에 떨어질 때까지 반복해 path를 구했다.