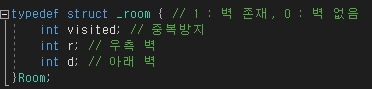
12주차 결과보고서

전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**





위와 같은 구조체와 전역변수들을 사용하여 .maz파일을 화면에 그리는 기능을 구현하였다.

Maze : 미로 데이터를 저장하는 배열의 주소를 저장하는 포인터

( 2차원 Room 타입 배열을 동적 할당받아 미로 데이터를 저장한다.)

Row, Col : 각각 미로의 행(세로), 열(가로) 크기를 저장한다.

baseX, baseY : 각각 미로를 그릴 시작점의 X, Y좌표를 나타냄

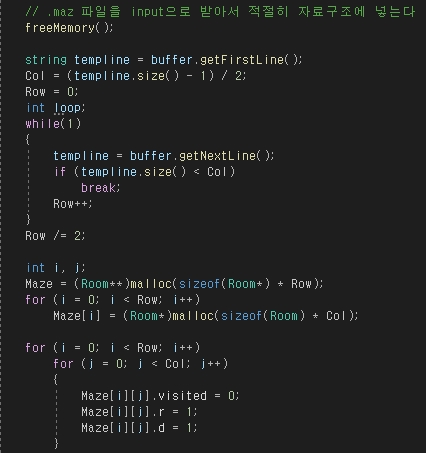
Length : 미로의 칸당 길이를 나타냄

visited : (BFS 및 DFS전용) 해당 방을 방문했는지를 저장함 (1 : 방문함, 0 : 방문안함)

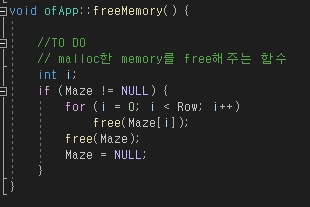
r : 우측 벽의 존재 여부 (1 : 있음, 0 : 없음)

d : 아래 벽의 존재 여부 (1 : 있음, 0 : 없음)







****

File - Open으로 .maz파일을 불러오면, readFile()에서 ofBuffer 클래스 타입 변수 buffer에 불러온 파일의 데이터를 저장하고 freeMemory()를 호출해 이전에 할당된 미로 데이터들을 모두 할당 해제시킨다. (이전에 할당된 메모리가 없다면 Maze가 NULL이므로 넘어감)

templine에 buffer.getFirstLine()의 값을 저장해 파일의 첫번째 줄을 가져온다.

미로 데이터의 가로 길이가 N이면 한 줄에 출력되는 +와 –의 개수의 합은 2N+1개이므로

Col = (templine.size()-1)/2로 미로의 가로 크기를 저장한다.

그 다음, buffer.getNextLine()을 계속 실행하다가 EOF(파일의 끝)에 도달하면 해당 줄의 길이가 0이 되므로 (templine.size()가 0이됨) 탈출할 때까지 Row를 0부터 1씩 증가시키고 루프를 탈출한 다음 2로 나눠 최종적으로 Row에 미로의 세로 크기가 저장되게 한다.

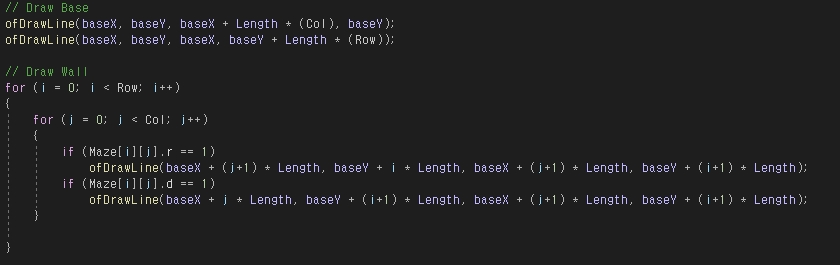
(세로 크기가 M인 경우, 파일에 출력된 전체 줄 수는 2M+1개이고, 처음 가로 크기를 측정하면서 1줄을 넘겼고, 루프문 속에서 탈출할 때까지 정확히 2M번을 루프했으므로 탈출 후 Row를 2를 나누면 미로의 세로 크기가 저장되게 된다.)

그런 다음 앞에서 구했던 미로의 가로, 세로 크기를 사용해 Maze에 2차원 배열을 할당받고, 할당 받은 미로 데이터를 초기화한다. (모든 벽이 있는 상태로 초기화 함)

미로의 벽 정보를 읽기 위해 buffer.getFirstLine()를 사용한다.

위에서 2번째 줄부터 마지막 줄까지 차례대로 한번에 2줄씩 읽어서 각각 우측 벽과 아래 벽의 문자를 읽어 각 방의 r, d의 값을 갱신한다. (rline : 짝수 번째 줄, dline : 홀수 번째 줄) rline의 2j+2번째 문자가 ‘ ‘면 Maze[i][j]의 우측 벽이 열린 상태이고, ‘|’면 닫힌 상태이다. (Maze[i][Col-1]의 우측 벽은 미로의 테두리이므로 항상 1임)

dline의 2j+1번째 문자가 ‘ ‘면 Maze[i][j]의 아래 벽이 열린 상태이고, ‘-‘면 닫힌 상태이다. (Maze[Row-1][j]의 아래 벽은 미로의 테두리이므로 항상 1임)



ofApp::draw()에서 위에서 Maze에 저장한 미로 데이터를 사용해 미로를 그린다.

먼저, 배열에 따로 저장되어 있지 않은 벽인 미로의 위쪽 테두리와 왼쪽 테두리를 그린다.

그 다음 2차원 배열을 2중루프문으로 모든 원소를 순회하면서 해당 원소의 r이 1이면 (우측 벽이 있으면) 우측 벽을 그리고, d가 1이면 (아래 벽이 있으면) 아래 벽을 그린다.

미로 데이터를 저장하는데 2차원 배열을 사용했으므로 미로의 가로 크기를 Col, 세로 크기를 Row라 하면, 공간복잡도는 O(Row\*Col) 이고, 2중 루프문을 사용했으므로 시간복잡도는 O(Row\*Col) 이다.

실험 전에는 전역변수 int Init = 0을 이전에 파일을 열었던 경우와 처음 파일을 여는 경우를 구분하여 freeMemory()의 호출 여부를 결정하는 용도로 사용하려고 했으나, 실험에서는 코드 간소화를 위해 Init을 사용하는 대신, Maze에 할당된 메모리를 해제할 때 Maze = NULL을 추가하고, freeMemory() 호출시 Maze가 NULL이 아닐 때만 Maze에 할당된 메모리를 해제하는 방식으로 변경하였다.

**2.**

이번 실습을 진행하면서 openframeworks 에서 ofxWinMenu 모듈을 사용하여 윈도우 창에 메뉴를 달고, 메뉴 내부에 팝업 메뉴와 분리선을 삽입하는 방법에 대해서 자세하게 알게 되었다. 또한, readFile()을 구현하면서, openframworks의 ofBuffer 클래스 타입 자료구조와 메서드(getFirstLine(), getNextLine() 등)를 어떻게 다뤄야 하는지 알게 되었다.

마지막으로, 프로그램 내부에서 임의의 개수의 파일을 열고 닫는 기능을 구현할 때는 파일을 교체해줄 때 마다 해당 파일에 관련된 정보를 저장하기 위해 할당된 메모리들을 반드시 해제해줘야 한다는 것을 알게 되었다. (해제하지 않는 경우 파일을 열고 닫을 때마다 계속 메모리 누수가 누적되므로 반드시 해제 해줘야함)