12주차 결과보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181294 이름: 임승섭

**1.**

우선 미로의 모양을 저장할 2차원 배열을 이용하였다. 배열의 row와 column의 크기는 각각 미로의 가로, 세로의 크기로 저장하였다. 이 때, 여기서 의미하는 가로와 세로의 크기는 미로 내의 방의 개수를 의미하는 것이 아닌, 모든 벽들을 포함한 문자의 개수를 의미한다. 즉, 실제 가로의 방이 n개가 있었다면, WIDTH에 저장되는 값은 2\*n + 1 이 된다. 세로 HEIGHT도 마찬가지이다.

2차원 배열 input[HEIGHT][WIDTH] 의 원소들은 readFile() 함수 내에서 file을 읽어와서, 파일 내에 있는 미로의 원소들을 그대로 받아왔다. 메모리는 new 를 이용하여 동적 할당 해주었고, 나중에 freeMemory() 함수 내에서 delete[] 를 이용하여 해제시켜주었다.

실제 미로를 화면에 그릴 때에는 모두 input 배열 내의 원소들을 이용하였다. 이중 for문을 통해 원소에 접근하였고, if문을 이용해 원소가 어떤 문자인지를 확인하여 각 문자마다 다른 코드를 작성하였다. 우선 크게 i(높이)가 짝수인 경우와 홀수인 경우로 나누었다. 짝수인 경우는 수직 경로를 막는 벽이 존재하는 부분이고, 홀수인 경우는 미로의 길, 즉 방이 존재하는 부분이다.

따라서 우선 짝수인 경우에는 존재할 수 있는 문자가 ‘+’, ‘-‘, ‘ ‘가 있다. ‘+’인 경우에는 굳이 이름을 주자면 방의 꼭짓점을 의미하기 때문에, 실제 미로를 그릴 땐 따로 구현할 내용이 없다. 하지만 각 행의 마지막 부분의 꼭짓점에 도달하면, curr\_x(현재 위치의 x좌표)를 다시 맨 왼쪽으로 돌려주어야 하기 때문에 curr\_x = 0 을 추가해주었고, 행의 마지막 부분이 아닐 때에는 continue를 추가하여 바로 다음 loop으로 넘어가도록 하였다. ‘-‘인 경우에는 실제 벽을 그대로 그려주면 된다. 벽을 그려준 이후에는 curr\_x(현재 위치의 x좌표)를 그려준 선의 길이만큼 오른쪽으로 이동시켜서 현재 위치를 바꿔주어야 한다. 그래서 ofDrawLine 함수를 이용하여 (curr\_x, curr\_y)에서 (curr\_x + length, curr\_y)까지 선을 그리도록 하였고, curr\_x += length를 추가하여 현재 위치를 옮겨주었다. ‘ ‘인 경우에는 수직 경로가 뚫려있는, 즉 벽이 없는 곳이기 때문에 curr\_x만 오른쪽으로 옮겨주면 된다. curr\_x += length 를 추가하여 현재 위치만 옮겨주었다.

홀수인 경우에는 존재할 수 있는 문자가 ‘|’, ‘ ‘가 있다. ‘|’인 경우에는 수평 경로의 벽이 존재하는 부분이므로 그대로 벽을 그려주면 된다. 마찬가지로 curr\_x도 오른쪽으로 이동시켜 다음 작업을 수행할 수 있도록 한다. 만약 행의 마지막 벽이라면, 현재 위치를 맨 왼쪽으로 옮겨주어야 한다. 이 때, curr\_x도 옮겨주어야 하지만, curr\_y도 수정하여 다음 행의 작업을 수행할 수 있도록 해야 한다. 즉, loop을 돌면서 현재 위치를 다음 행으로 넘겨주는 작업은 이 부분에서 수행된다. ofDrawLine함수를 이용해서 (curr\_x, curr\_y)에서 (curr\_x, curr\_y + length)까지 선을 그려주었고, curr\_x += length 를 통해 x좌표를 오른쪽으로 한 칸 옮겨주었다. if(j == WIDTH-1) 조건문 내에 들어왔을 때는 curr\_y += length 를 통해 y좌표를 아래로 한 칸 내려주었고, curr\_x = 0 을 통해 x좌표를 맨 왼쪽으로 초기화시켰다. ‘ ‘인 경우에는 미로의 방을 의미하는 곳일 수도 있고, 수평 경로가 뚫려있는 부분을 의미하는 곳일 수도 있다. 그래서 j가 짝수인 경우와 홀수인 경우로 나누어서, 짝수인 경우가 벽이 뚫려있는 곳, 홀수인 경우가 미로의 방 이라고 고려하였다. 사실 미로의 방인 경우에는 따로 작업할 내용이 없고, 수평 경로가 뚫려있는 부분일 때에는 curr\_x += length를 통해 x좌표를 이동시키는 작업을 수행하도록 하였다.

미로를 그리는 draw() 함수 내에서는 조건문은 많지만 사실 loop은 초기에 설정해둔 미로의 가로, 세로를 탐색하는 이중 loop이 전부이다. 즉 시간복잡도는 O(WIDTH \* HEIGHT)이다. 또한 메모리도 readFile()함수에서 동적 할당으로 생성한 2차원 배열 input[]이 전부이기 때문에 공간복잡도는 O(WIDTH \* HEIGHT)이다.

실제로 작성한 코드는 예비보고서에 작성했던 pseudo code와 크게 다른 점이 없다. 다만 행의 마지막 부분인지 확인하는 조건문 if(j == WIDTH - 1) 이 정확한 위치에 있지 않아서, 실제 구현할 때에는 화면에 미로가 그려지는 것을 보면서 수정하였다. 이 외에는 실험 전에 생각했던 그대로 구현할 수 있었다.

**2.**

완전히 처음 겪어본 것은 윈도우 창의 버튼이 구현되는 점이었다. 사실 이미 코드에 다 작성되어 있었기 때문에 직접 구현하진 않았지만, 프로그램을 실행했을 때 윈도우 창에 버튼이 존재하고, 누를 수 있다는 점에 호기심이 생길 수밖에 없었다. AddPopupMenu() 함수나 AddPopupItem() 함수 등을 보고 어떤 방식으로 구현되는지 전반적인 지식을 쌓을 수 있었다. 아마 최종 프로젝트를 할 때 이 부분을 실제로 작성하여 구현해볼 생각이다.