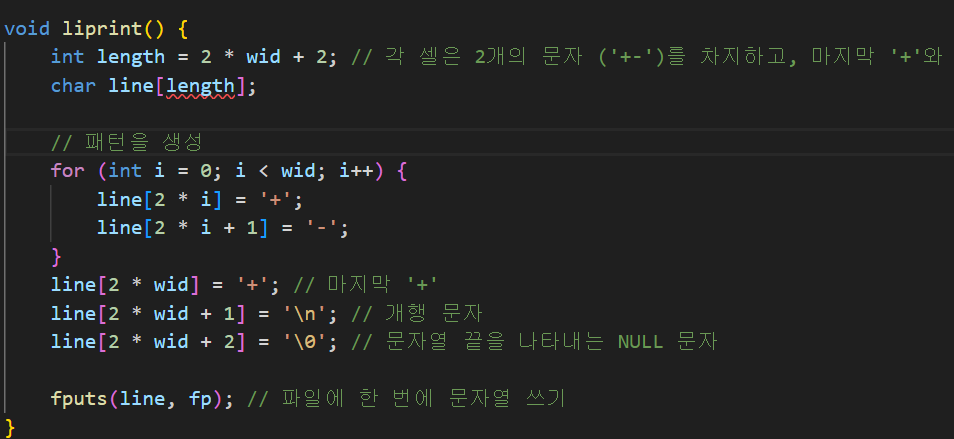
11주차 결과보고서

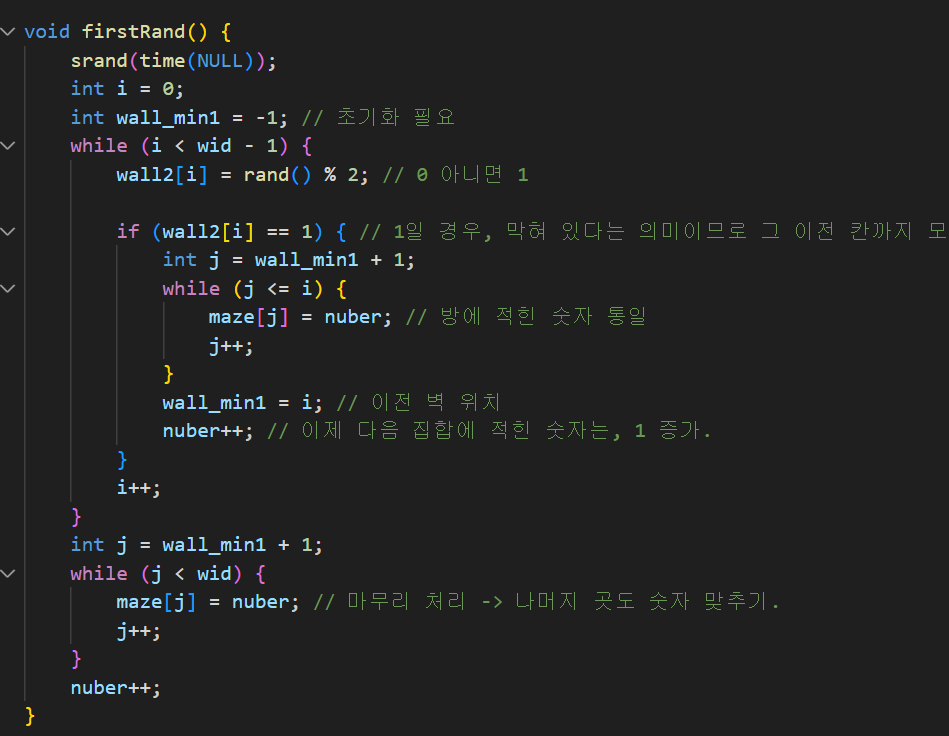
전공: 컴퓨터공학과 학년: 4학년 학번: 20191612 이름: 윤기웅

**1. (문제 및 제목은 작성하지 말 것. 답안만 작성하면 됩니다.)**

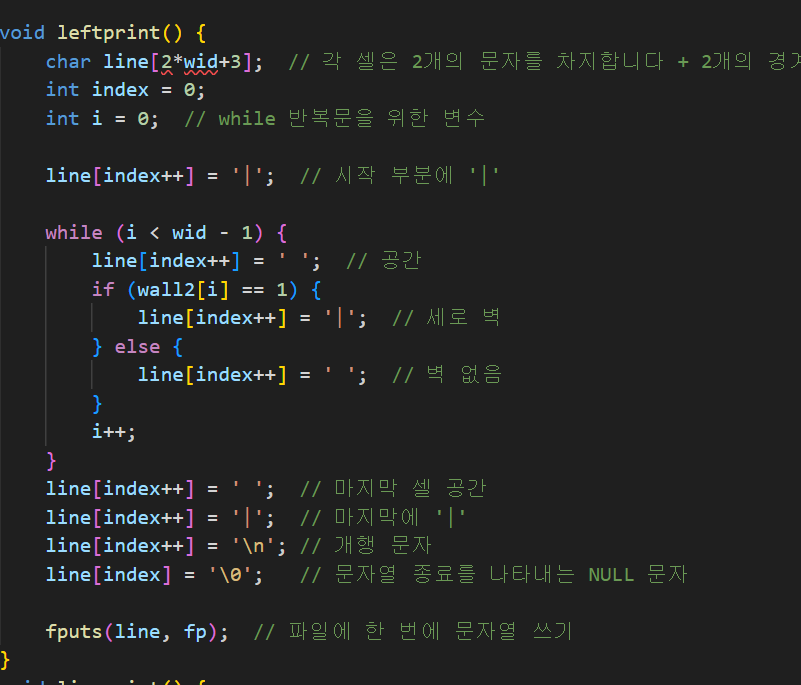
.................



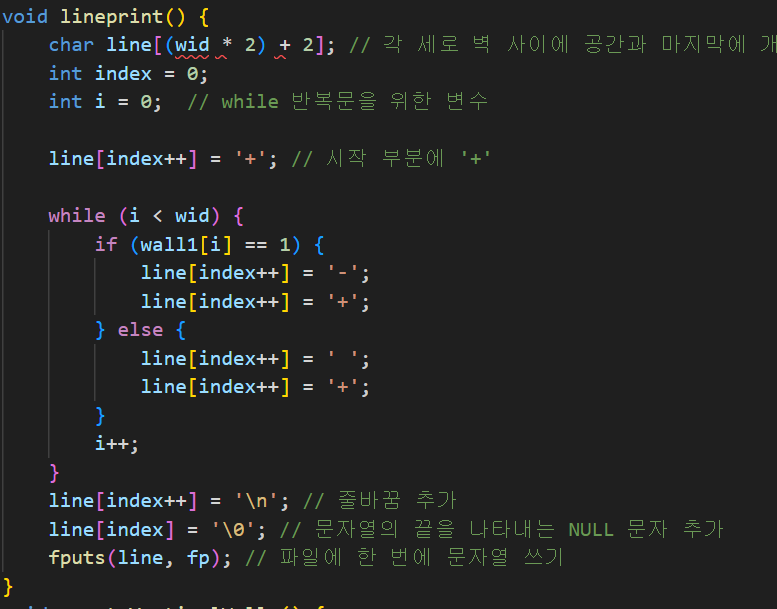
첫 번째 함수인 liprint는 미로의 상단 또는 하단 경계를 출력하는 역할을 한다. 각 셀은 '+-'로 표현되며, 마지막에는 '+'가 추가되고 개행 문자('\n')로 끝난다. 이 함수는 wid라는 너비 변수를 기반으로 하여, 미로의 가로 길이에 맞는 경계선을 생성하고 이를 파일에 쓰는 역할을 한다. line이라는 문자 배열을 사용하여 경계선 패턴을 만들고, fputs 함수를 통해 이를 한 번에 파일에 쓰기 위해 사용한다.



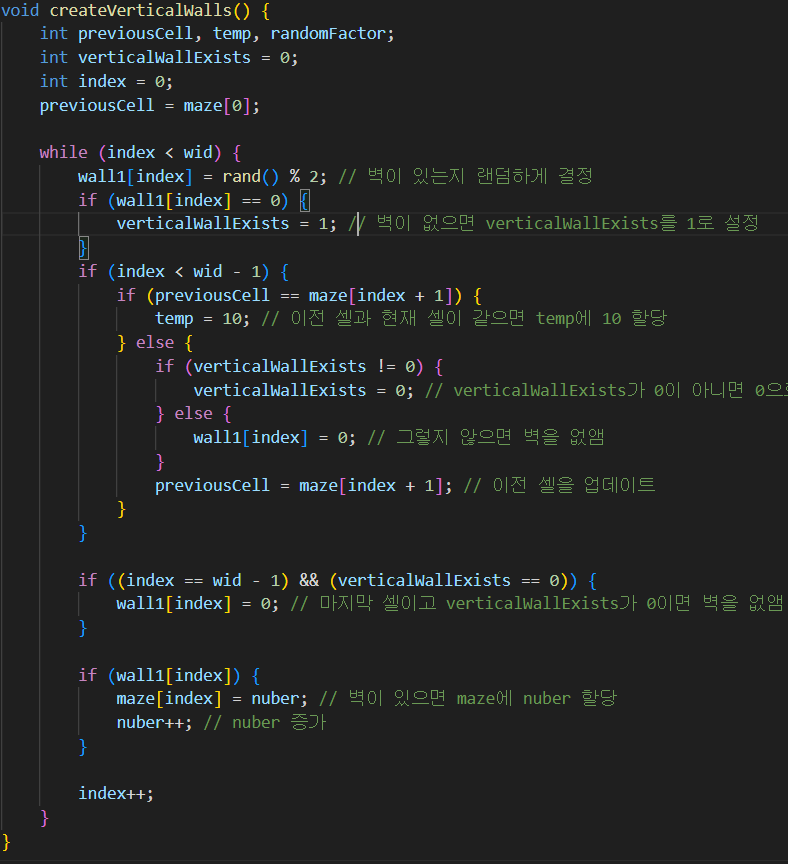
두 번째 함수인 firstRand는 미로의 수평 벽을 무작위로 생성하는 역할을 한다. rand 함수를 사용하여 0 또는 1의 값을 생성하고, 이 값에 따라 벽을 설정한다. 1이 나오면 해당 위치에 벽을 세우고, 이전 벽 위치부터 현재 위치까지 같은 번호(nuber)를 부여하여 같은 방으로 처리한다. 이 과정을 통해 미로 내에서의 방을 구분하고, 각 방에 고유 번호를 부여하여 미로 생성의 다양성을 제공한다. 이 함수는 미로 생성의 첫 단계에서 수평 벽을 설정하는 데 중요한 역할을 한다.



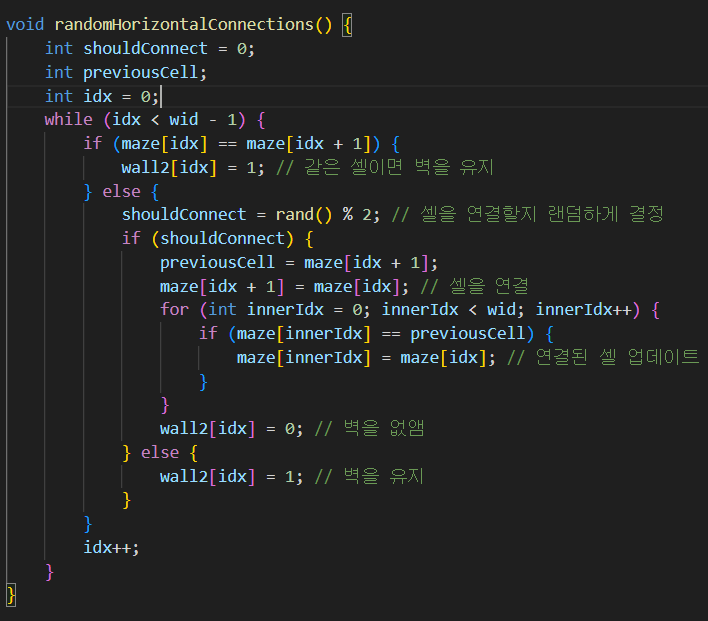
세 번째 함수인 leftprint는 미로의 세로 벽을 출력하는 역할을 한다. 이 함수 역시 wid 변수를 기반으로 하여, 각 셀 사이에 벽이 있는지 없는지를 판단하여 파일에 쓰게 된다. 벽이 있을 경우 '|'를, 없을 경우 ' '를 삽입하여 미로의 세로 구조를 표현한다. 마지막에는 항상 '|'를 추가하여 미로의 우측 경계를 표현하고, 개행 문자를 통해 다음 줄로 넘어갈 준비를 한다. 이 함수는 미로의 수직 구조를 파일에 출력하는 데 사용된다.



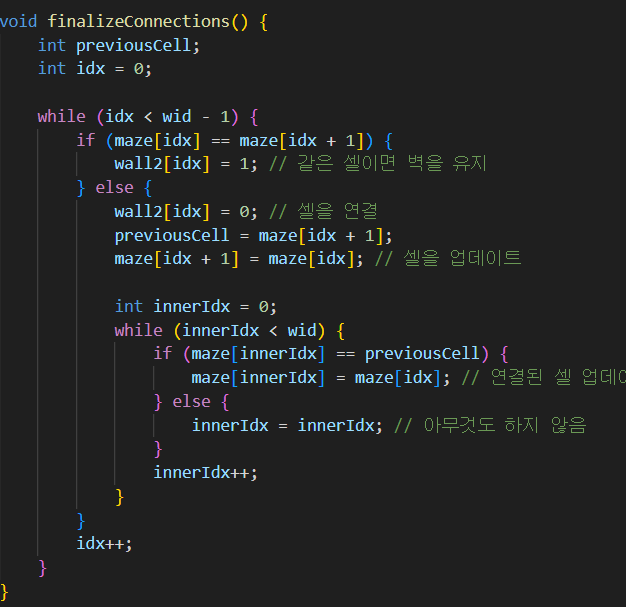
다음 함수인 lineprint는 미로의 수평 벽을 표시하기 위한 함수다. 이 함수는 wid 변수를 사용하여 미로의 너비를 기준으로 한 줄을 구성한다. 각 셀 사이의 수직 벽의 존재 여부에 따라 '-' 또는 공백을 추가하고, 각 셀의 끝에는 '+'를 추가하여 셀의 구분을 나타낸다. 줄의 마지막에는 개행 문자를 추가하여 줄바꿈을 하고, 이렇게 완성된 문자열을 파일에 쓴다.



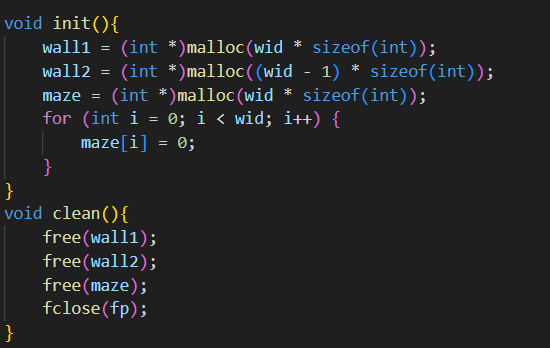
다음 함수인 createVerticalWalls는 미로의 수직 벽을 생성하는 함수다. 이 함수는 미로의 각 셀에 대해 랜덤하게 벽의 존재 여부를 결정한다. 만약 인접한 셀이 같은 값을 가지고 있다면, 이를 기반으로 수직 벽의 존재 여부를 결정하여 미로의 연결성을 조절한다. 이 과정에서 nuber 변수를 사용하여 새로운 셀 값을 할당하며, 이는 미로 내에서 고유한 셀을 식별하는 데 사용된다.



다음 함수인 randomHorizontalConnections는 미로 내에서 수평 연결을 랜덤하게 생성하거나 제거하는 역할을 한다. 이 함수는 연속하는 셀들 사이의 연결성을 랜덤하게 결정하여 미로 내에서 경로의 다양성을 늘린다. 만약 인접한 셀이 다른 값을 가지고 있을 경우, 랜덤하게 이들 셀 사이의 벽을 제거하여 두 셀을 연결시키거나, 벽을 유지하여 셀들 사이를 분리시킨다.



finalizeConnections 함수는 미로 내에서 수평 연결을 최종적으로 확정짓는 역할을 한다. 이 함수는 배열 maze의 인접한 셀을 순회하면서, 만약 인접한 셀의 값이 같다면 wall2 배열에 1을 할당하여 벽을 유지시킨다. 반면 인접한 셀의 값이 다를 경우, wall2 배열에 0을 할당하여 벽을 제거하고 셀을 연결한다. 이후, 연결된 셀의 값을 업데이트하기 위해 전체 미로를 다시 순회하며, 이전 셀의 값을 가진 모든 셀을 현재 셀의 값으로 업데이트한다. 이 과정은 미로의 경로를 단순화하고 연결된 구조를 만드는 데 기여한다.



init 함수는 미로 생성을 위한 초기 설정을 담당한다. 이 함수에서는 wall1, wall2, 그리고 maze 배열을 동적 메모리 할당을 통해 생성한다. wall1은 미로의 수직 벽을 나타내며, wall2는 수평 벽을 나타낸다. maze 배열은 미로의 각 셀을 나타내며, 모든 셀의 초기 값은 0으로 설정된다. 이 초기화 과정은 미로 생성 알고리즘을 실행하기 전에 필수적인 단계이다.

clean 함수는 미로 생성 후에 사용했던 자원을 해제하는 역할을 한다. 이 함수에서는 malloc을 통해 할당받은 wall1, wall2, maze 배열의 메모리를 해제하고, 열려 있는 파일 포인터 fp를 닫는다. 이는 프로그램이 할당한 자원을 적절히 관리하고 메모리 누수를 방지하는 데 필요한 정리 작업이다.

이 코드는 미로 생성 알고리즘 중 하나인 Eller's Algorithm을 구현한 것이다. Eller's Algorithm은 미로를 생성할 때 행(row) 단위로 순차적으로 진행하며, 각 행에서 셀들을 집합으로 관리하여 미로를 생성한다. 이 알고리즘은 특히 큰 미로를 생성할 때 메모리 사용량을 줄일 수 있는 장점이 있다. Eller's Algorithm은 행 단위로 미로를 생성하기 때문에, 전체 미로를 메모리에 저장할 필요가 없고 이는 큰 미로를 생성할 때 매우 유용하며, 메모리 사용량을 크게 줄일 수 있다. 다른 미로 생성 알고리즘들, 예를 들어 재귀적 백트래킹(Recursive Backtracking)이나 Prim's Algorithm, Kruskal's Algorithm과 비교했을 때, Eller's Algorithm은 현재 처리 중인 행만을 메모리에 유지하면 되기 때문에 공간복잡도 측면에서 이점을 가진다.

또한, Eller's Algorithm은 각 행을 처리할 때 집합을 이용해 셀들을 관리하기 때문에, 미로의 수평 방향 연결성을 보장하며 다양성을 제공할 수 있다. 이는 미로의 구조가 더 다양해지고 복잡해질 수 있게 한다. 다른 알고리즘들에 비해 이러한 특성 때문에 Eller's Algorithm은 미로 생성에 있어서 특히 유연하고 효율적인 방법 중 하나다.

**시간복잡도**

이 코드의 시간복잡도를 분석해 보면, 주요 함수들(firstRand, createVerticalWalls, randomHorizontalConnections, finalizeConnections)은 모두 wid에 비례하는 반복문을 포함하고 있고 leftprint와 lineprint 함수 또한 각각 wid에 비례하는 반복문을 포함하고 있다. 메인 함수에서 이러한 함수들을 hei만큼 반복해서 호출한다. 따라서, 전체 시간복잡도는 O(wid \* hei)로 요약할 수 있다.

**공간복잡도**

공간복잡도 측면에서, 이 코드는 maze, wall1, wall2라는 세 가지 주요 배열을 사용한다. maze와 wall1은 wid에 비례하는 공간을 차지하고, wall2는 wid-1에 비례하는 공간을 차지한다. 따라서, 전체 공간복잡도는 O(wid)로 요약할 수 있다.