12주차 결과보고서

전공 : 컴퓨터공학과 학년 : 2학년 학번: 20211558 이름 : 윤준서

**1. 자료구조 설명**

텍스트, 스크린샷, 폰트, 대수학이(가) 표시된 사진

자동 생성된 설명

'-'과 '|'을 선으로 만들어 저장해 놓은 LINE 벡터를 반복문을 통해 화면에 선을 그리는 함수 maze\_line을 이용하여 선을 출력한다. 예비 보고서에서 작성한 초안과 동일하다. 해당 함수의 시간 복잡도는 LINE의 원소 개수가 n일 때, O(n)이고, 공간 복잡도 또한 O(n)이다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

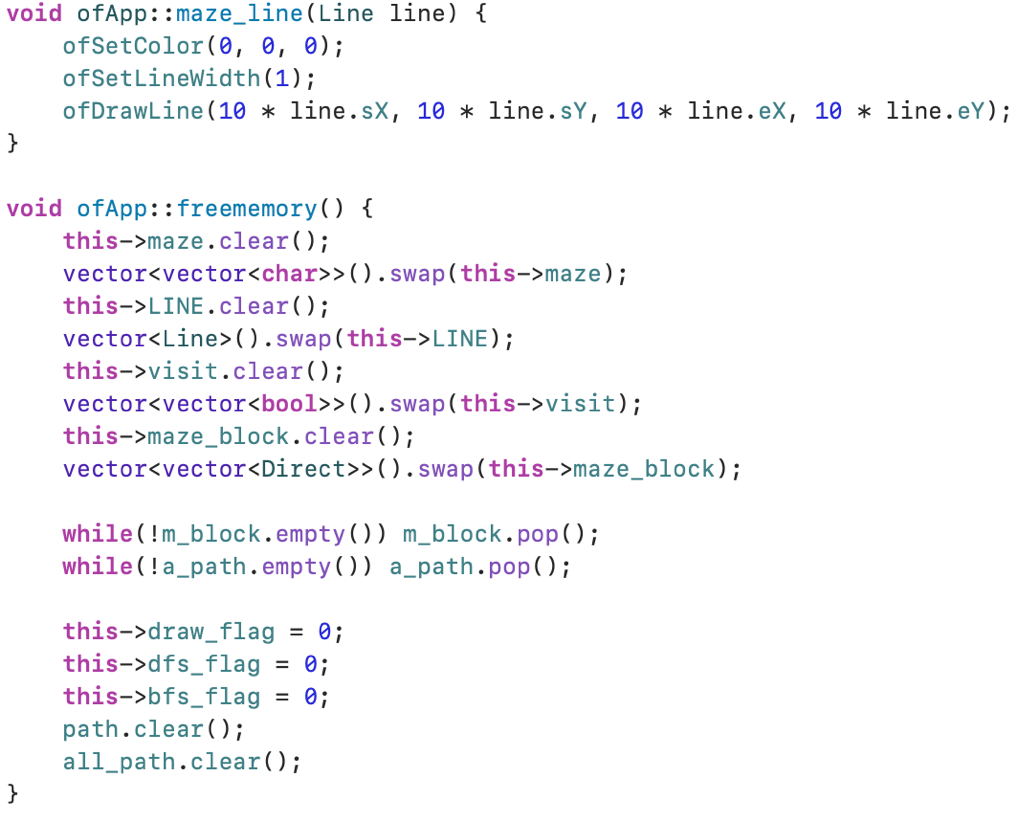
자동 생성된 설명

자료들을 정의하기 전 메모리 누수 방지를 위해 freememory()를 실행한다. maz file을 읽는 변수를 두 개를 설정한다. 하나는 그려야 할 벽들의 정보를 저장하기 위해 만든 배열 maze의 크기를 측정하기 위해 WIDTH, HEIGHT를 계산하는 용도이고, 하나는 계산한 인덱스 만큼 파일의 '-'과 '|'문자를 읽어 그려야 할 선들의 시작 좌표와 끝 좌표를 line\_temp에 저장한다. 그리고 이 line\_temp를 LINE벡터에 하나의 원소로 저장하는데, maze\_line함수를 이용해 선을 화면에 출력할 때 용이하다.

처음 maze[][]의 인덱스를 계산할 때 벽과 벽 사이의 모든 '+'를 생략해야 하므로 (WIDTH - 1) / 2, (HEIGHT - 1) / 2 으로 계산한다. 이후 두 번째로 파일을 읽을 때 maze[][]에 '-'과 '|'의 정보를 순서대로 저장한다. 이들을 토대로 선의 정보를 만들어 line\_temp변수에 선의 시작 좌표와 끝 좌표를 저장하는데, 이 때 정수배에 맞게 좌표를 계산해야 화면 출력 시 선들의 간격이 이탈되지 않고 일정하다.

이 때 인덱스 i, j에 따라 좌표가 정해지는데, 이후 maze\_line함수에서 좌표의 숫자 \* 10 만큼 다시 조정하여 선을 그리게 된다. maze\_line 함수를 반복문으로 호출하기 위해 line\_temp변수를 벡터 LINE에 순서대로 저장한다. 예비 보고서의 초안과 동일하다.

해당 자료구조의 시간 복잡도와 공간 복잡도 모두 O(WIDTH \* HEIGHT)이다.



maze\_line()은 선의 색과 굵기를 정하고 Line구조체에 저장된 선의 시작 좌표와 끝 과표를 받아 10을 곱한 만큼의 좌표에 선을 출력한다. 즉 선의 길이가 10이 된다. 예비 보고서의 초안과는 선의 길이만 다르다. 시간 복잡도와 공간 복잡도는 Line구조체 하나만을 가지고 계산하므로 모두 O(1)이다.

freememory()는 구조체와 변수 자료들을 초기화한다. 벡터를 초기화할 때 clear()과 swap()을 사용하는데, clear()는 벡터의 요소들을 해제하고, swap()은 벡터가 할당한 메모리를 해제한다.

스택을 초기화할 때는 스택이 빌 때까지 pop()을 실행한다.

int 자료형을 초기화할 때는 0으로 초기화한다.

마지막으로 경로 벡터를 clear()만 이용하여 해제한다.

해당 자료구조의 시간 복잡도는 벡터와 스택의 크기에 영향을 받는다. 6개의 벡터의 각 크기가 a, b, c, d, e, f고 2개의 스택의 각 크기가 g, h이면 전체 시간 복잡도는 O(a+b+c+d+e+f+g+h)가 된다. 공간 복잡도의 경우 상수 공간 외에 별다른 공간이 없으므로 O(1)이다.

**2. 본 실험을 하면서**

익숙한 C언어를 통해 만든 미로를 익숙하지 않은 C++언어를 통해 구조체를 만들 때, 기존에 C언어에 없는 vector, swap, push\_back 등의 C++의 내장된 클래스, 함수에 대해 자세히 조사하고 코드를 작성하면서 각각의 역할들이 무엇인지 더 확실하게 알게 되었다. 처음엔 코드를 에러가 없을 정도로만 자유롭게 작성하였는데, 여러 함수들을 이용해 최적화를 반복하면서 코드를 더 간결하게 만들 수 있었다.

선을 어떻게 표현할 지 구상을 하다 2차원 배열에 차례로 '-'와 '|'를 openframeworks의 선으로 하나씩 만들어 저장하는 방식을 택했는데, 단순한 만큼 가시성이 좋아서 실수로 에러가 난 부분들을 빠르게 찾아 수정할 수 있었다.