자료구조 과제 1 보고서

04분반 20222663 권기영

1. **문제 내용**

문제 내용은 일정한 규칙이 있는 주어진 미로 텍스트 파일을 적은 데이터로 저장한 뒤 다시 복원하는 것이다. 이 문제의 핵심은 규칙이 있다는 것이고, 이 규칙을 활용하여 적은 데이터지만 모든 미로를 저장한 것과 같은 의미를 지니는 데이터를 저장해야 한다.

1. **문제 해결**
2. **미로 저장**

미로를 저장하는 데 있어서 생각했던 핵심 아이디어는 ‘+’, ‘|’, ‘-‘ 3개 모양의 벽들 중 ‘+’ 모양의 벽의 정보만 저장하면 된다고 생각하였다. 3가지 모양의 벽 전체를 모두 저장하게 된다면 벽 하나에 들어가는 정보의 양이 (x좌표, y좌표, 모양) 총 3byte인데다가 벽 전체를 저장하기 때문에 byte수가 매우 커지게 된다. 그러나 ‘+’만 저장하고도 같은 정보를 모두 저장할 수 있다. ‘+’의 x좌표, y좌표, 뻗어 나가는 방향(오른쪽에 ‘-‘가 있으면 오른쪽으로 뻗어 나가는 것, 아래쪽에 ‘|’가 있다면 밑으로 뻗어 나가는 것.)에 대한 정보를 저장하면 하나의 ‘+’ 당 3byte를 사용하지만 ‘+’에 대한 정보만 저장하기 때문에 벽 전체를 저장하는 것보다 효율적이다.

그러나 아무리 생각해봐도 정보 하나당 3byte를 사용하는 것은 너무 비효율적이라고 생각하였다. 그래서 1차원 배열만을 활용하여 ‘+’에 대한 정보를 저장할 수 있다는 사실을 알게 되었다.

첫 번째로, y좌표의 정보를 따로 1바이트씩 할당하는 것이 아니라, 한 행이 끝나면 행이 끝났다는 사실을 알려주는 정보를 저장해서 y좌표를 표현하는 것이다. 예를 들어 1maze의 경우 low가 12이기 때문에 12byte를 필요로 한다.

두 번째로, 뻗어 나가는 방향에 대한 정보를 따로 바이트 수를 할당할 필요가 없다. 이는 미로의 규격이 20 \* 20인 데에서 착안한 아이디어이다. 규격이 20 \* 20이기 때문에 x좌표 또한 0 ~ 19까지만으로 표현할 수 있다는 것인데, 알파벳의 수는 이보다 큰 26개임을 이용하였다. 뻗어 나가는 경우의 수가 오른쪽, 밑, 둘 다 총 3가지인데, 오른쪽으로 뻗을 경우 x좌표에 -1을 곱해서 저장, 밑으로 뻗을 경우 소문자 알파벳으로 변환하여 저장(0이 a, 1이 b …), 둘 다 뻗을 경우 대문자 알파벳으로 변환하여 저장, 뻗지 않을 경우에는 원래 x좌표를 저장하는 식으로 저장하였다.

또한 미로의 각 꼭짓점 4개에는 ‘+’가 있는 규칙이 있으므로 이 또한 제외할 수 있었다.

지금까지 핵심 아이디어에 대한 설명이었고, 코드에 대해서 순차적으로 설명을 하도록 하겠습니다.

**<코드 설명>**

1. 우선 텍스트 파일을 받아 열어서 열 값과 행 값을 int로 받는다(8 byte).
2. 그 후 끝까지 돌면서 ‘+’의 개수를 count로 세 주고 파일을 닫는다. 이 때, 마지막 행의 ‘+’개수는 세지 않는데, 그 이유는 마지막 바로 전 행에서 ‘|’가 나오면 그 밑이 ‘+’이기 때문에 굳이 저장하지 않아도 표현할 수 있기 때문이다.
3. 열(column), 행(row), ‘+’의 개수에 대한 정보를 받았으니 char형 1차원 배열 plus를 필요한 만큼 동적 할당 해준다. 그 크기는 sizeof(char) \* (count – 2) + row이다. 맨 위 행에서 꼭짓점은 무조건 ‘+’이기 때문에 count – 2이다. row의 크기는 y좌표를 대신하여 행이 끝날 때 마다 저장해야 하는 정보가 row개 있기 때문이다.
4. 파일을 다시 열고, 열과 행 줄을 건너뛴 뒤, 파일의 끝까지 가면서 ‘+’에 대한 정보를 plus에 저장한다. temp에다가 fgetc로 하나씩 저장하면서, 만약 temp == ‘+’라면 x좌표를 저장한 뒤, 오른쪽과 밑 두 가지의 정보를 받는다. 오른쪽은 fgetc를 한번 더 사용해서 정보를 받고, 밑 쪽은 fseek함수를 활용하여 한 줄을 뛰어넘어 입력을 받는다. 정보를 모두 받으면 다시 fseek함수를 활용하여 제자리로 돌아온다. 행이 끝나면 plus에 ‘x’(행이 끝났음을 알리는 정보)를 저장한다.
5. 오른쪽과 밑 두 가지의 정보를 받으면서, 오른쪽만 뻗어 나가면 x좌표 \* (-1) 로 저장, 밑으로만 뻗어 나가면 x좌표 + ‘a’로 저장, 둘 다 뻗어 나가면 x좌표 + ‘A’로 저장, 뻗어 나가지 않으면 x좌표만 그대로 저장한다.
6. 잘 저장되었나 확인하기 위해 저장된 ‘+’ 정보의 index값과 저장된 정보의 형태를 출력하여 확인한다.

maze1은 데이터 78byte, 규격 8byte로 86byte를 할당했고, maze2는 데이터70byte, 규격 8byte로 78byte를 할당하여 저장했다.

1. **미로 표현(draw\_maze)**

미로 표현에서 가장 중요한 것은 바로 미로의 규칙이다. 예를 들어 ‘|’의 오른쪽 왼쪽 옆에는 무조건 길이 있는 것과 같은 규칙을 이용하여 미로를 순차적으로 그려 나가야 한다. 또한 미로를 표현하는 것에서 중요한 점은 미로 표현에 사용되는 자료의 byte수는 상관이 없다는 점이다. 따라서 최대 규격 만큼 저장 공간을 할당하고, 원래 저장 했던 미로 데이터를 2차원 배열에 새롭게 저장해서 표현에 사용한다면 더 수월 할 것이다.

draw\_maze 함수에 전달한 매개변수는 미로를 저장한 1차원 배열 plus(char \*), 행(int), 열(int) 이렇게 총 3가지를 전달했다.

**<코드 설명>**

1. 최대 규격인 20 \* 20에 맞춰, 41 \* 41의 크기를 가진 int형 2차원 배열 arr를 선언한다. arr에다가 미로를 전부 그릴 것이다. 일단 0으로 전부 초기화하여, arr요소의 값이 0이라면 아직 모양이 결정되지 않음을 조건으로 이용할 것이다.
2. 꼭짓점 4개에 ‘+’를 그리고, 왼쪽 벽과 오른쪽 벽에 ‘|’를 그린다.
3. 배열 plus를 끝까지 탐색하면서, 값에 따라 ‘+’의 (x + 1)좌표, (y + 1)좌표, (x+1, y+1)좌표에 있는 것들을 그린다. 예를 들어, plus의 값이 음수라면, (x+1)좌표에 있는 것은 ‘-‘를 그리고, 나머지는 공백을 그린다. (x+1, y+1)의 좌표에 있는 것은 길이므로 무조건 공백이다.

(+)(x + 1)

(y + 1) (x+1, y+1)



1. 윗 벽과 아래 벽을 그린다. arr 요소의 값이 ‘+’가 아니라면 ‘-‘을 저장한다. 맨 위의 벽에 +가 있으면 그 바로 밑에 줄에는 ‘|’가 있는 규칙을 활용하여 밑의 줄도 완성한다. 짝수 행의 요소는 공백(길)과 ‘|’밖에 없기 때문에 완성이 된다.
2. 벽의 옆에는 길이 있는 규칙을 활용하여, arr전체를 돌면서, 아직 모양이 확정되지 않은 요소들(값이 0인 요소들) 중에 ‘|’의 왼쪽 오른쪽 이나, ‘-‘의 위쪽 아래쪽에 공백을 그린다.
3. 길은 +로 시작해서 +로 끝나는 규칙을 이용해서, 값이 0인 요소들 중에 ‘-‘가 왼쪽에 있다면 ‘-‘를 그려주고, ‘|’가 위쪽에 있다면 ‘|’를 그려주고, 아무것도 없다면 길을 그려준다. 이미 위에서 많은 모양을 확정 시켜 놓았기 때문에 가능한 조건이다.
4. 마지막으로 밑 2줄을 그린다. 밑에서 2번째 줄은, 위에 ‘|’가 나오면 ‘|’로 그린다. 맨 밑의 줄은, 위에 ‘|’가 나오면 ‘+’로 그린다.
5. arr의 요소들을 매개변수로 받아 놓은 행과 열의 크기에 맞춰서 출력한다.

<maze 1 결과>

텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명

<maze2 결과>

텍스트이(가) 표시된 사진

자동 생성된 설명