**Report**

**For Data Structures**

**학과 :**

**학번 :**

**이름 :**

컴퓨터공학과

20184071

김도현

**문제 1**

* [교재 2장 연습문제 21]
  + 연습문제 21의 문제에서 주어진 그래픽을 2차원 배열로 표현하라. 그래픽에서 노랑색 셀은 2로 표현하고, 흰색 셀은 0으로 표현한다.
  + 위의 그래픽에서 흰색 셀을 검정색 셀(1로 표시)로 채우고자 한다.
  + 위의 그래픽에서 특정 흰색 셀이 주어졌을 때, 모든 흰색 셀 영역을 검정색 셀로 채우는 알고리즘 flood\_fill()을 순환 기법을 사용하여 작성하라.
  + 그래픽에서 채워야 할 영역셀이 오른쪽 그래픽과 같이 그 경계를 포함한 경우까지 고려하라.
  + 위의 알고리듬을 C 함수로 작성하고, main() 함수를 작성하여 테스트하고, 그 결과를 검증하라.
    - 테스트는 영역이 그래픽 경계를 포함하는 경우와 그렇지 않은 경우(교재에 제시됨)에 대해서 모두 수행하라.

1. 문제분석

|  |  |
| --- | --- |
| Input | Int x (x 좌표) , y ( y 좌표 ) , 수정전의 int 형 2 차원배열 screen[WIDTH][HEIGHT] |
| Output | 수정 후의 int 형 2차원배열 screen[WIDTH][HEIGHT] |
| Process | 1. X 좌표 y 좌표에 칠해진 색상이 무엇인지 판별하는 함수를 작성한다. 함수명은 screen\_read(int x, int y) 이다.   - x 랑 y 를 parameter 로 받고, return screen[x][y] 를 반환한다.  반환형태는 int 형이다.   1. X 좌표 y 좌표에 검정색으로 칠하는 함수를 작성한다. 함수명은 write\_pixel(int x, int y) 이다.   - x 랑 y 를 parameter 로 받는다. 그 후, screen[x][y] = 1 로 설정한다.  단지 수정만 하므로 반환형태는 void 형이다.   1. Int 형 2차원배열 screen 을 출력하는 함수를 작성한다. 함수명은 print\_screen() 이다.   - 2중 for 문을 사용하여 2차원배열의 width height 원소를 모두 출력한다.  단지 출력만 하므로, 반환 형태는 void 형이다.   1. Int 형 2차원 배열 screen 의 white 부분 모두 black 으로 칠하는 함수를 작성한다. 함수명은 flood\_fill(int x, int y) 이다.   - 문제에서 제시된 해결 방법이 순환을 이용하여 칠하는 것이다.  첫번째로, 순환문이 종료되기 위해선, 순환을 수행하면서 앞으로 더 수행되어야 할 횟수가 줄어들어야 한다. 그렇기 때문에 순환을 수행하면서, 칠해지지 않은 부분 (white) 가 줄어들어야 한다.  그렇기 때문에, 순환을 수행할 때 먼저 흰 부분을 색칠을 하고 색칠이 되었으면 연속되어 있는 부분을 칠해야 하기 때문에, 상 하 좌 우 모두 확인을 하고 색상을 칠하면 된다.  그러나, 흰 부분을 칠하기 전, 그 픽셀이 흰색인지 확인해야 한다. 왜냐하면 무턱대고 칠해버리면, 원래 있던 노란색부분도 검정색이 될 뿐더러, 순환문이 무조건적으로 반복될수 있기 때문이다.  위 내용을 조합해서 4번의 알고리즘은 다음과 같다.   1. X y 좌표가 흰색이면 수행을하고 흰색이 아니면 해당 함수는 수행하지 않는다. 2. 흰색일 경우면, 흰색인 부분은 검정색으로 칠해준다. 3. 검정색으로 칠했으면 연속적인 부분(상하좌우) 을 칠해야 하므로,   flood\_fill(x + 1 , y), flood\_fill(x – 1 , y), flood\_fill(x , y+1), flood\_fill(x , y – 1) 을 수행한다.   1. 하지만 이렇게 하면 문제가 생긴다. 맨 위쪽 경계면이 검정색일 경우, 해당 함수는 X 의 flood\_fill ( 0 – 1 , y ) 을 ( 맨위면 x 의 값은 0 이다 ) 수행하게 될것인데, 배열에선 –1 인덱스가 존재하지 않는다. 그렇기 때문에 x 또는 y 가 0 인경우면 flood\_fill 을 호출하지 말하야 한다. 마찬가지로 x 또는 y 좌표가 width 또는 height 를 넘겨 배열의 크기보다 큰 인덱스를 호출하면 오류가 생기므로, 해당 문제도 보정을 한다. |
| Example | 입력되는 케이스들은 다음과 같다.   1. 흰색이 경계면에 있는 경우와 시작하는 x y 좌표가 흰색인 경우 2. 흰색이 경계면에 있는 경우와 시작하는 x y 좌표가 노란색인 경우 3. 흰색이 경계면에 있는 경우와 시작하는 x y 좌표가 흰색이고, 다른 흰색이 연속되지 않는 경우 4. 흰색이 경계면에 없는 경우와 시작하는 x y 좌표가 흰색인 경우 5. 흰색이 경계면에 없는 경우와 시작하는 x y 좌표가 노란색인 경우 6. 흰색이 경계면에 없는 경우와 시작하는 x y 좌표가 흰색이고, 다른 흰색이 연속되지 않는 경우   각 항목에 대한 실행결과는 [3. 실행결과 ] 소주제에서 확인 가능하다. |

1. C언어를 사용한 프로그램 작성

#include <stdio.h>

#include<windows.h>

#define WHITE 0 //기호상수를 선언하여 혼동이 오지 않게 소스를 짠다.

#define BLACK 1

#define YELLOW 2

#define WIDTH 10

#define HEIGHT 10

int screen[WIDTH][HEIGHT] = //스크린인 임의의 2차원 배열을 생성한다.

//함수에서 접근하기 쉽게 하기 위해 글로벌 변수로 선언한다.

{

{0,0,0,0,0,0,2,2,2,2},

{2,2,2,2,0,2,2,2,2,2},

{2,2,2,0,0,0,0,2,2,2},

{2,2,2,2,0,0,0,2,2,2},

{2,2,2,2,0,0,0,2,2,2},

{2,2,2,2,0,0,0,0,2,2},

{2,2,2,2,0,2,2,2,2,2},

{2,2,2,2,0,2,2,2,2,2},

{2,2,2,2,0,2,2,2,2,2},

{2,2,2,2,2,2,2,2,2,2} };

int screen\_read(int x, int y); //x y 좌표 픽셀값 보는 함수 원형 정의

void write\_pixel(int x, int y); //x y 좌표 픽셀 검은색으로 칠하는 함수 원형 정의

void print\_screen(); //screen 을 모두 볼수 있는 함수 원형 정의

void flood\_fill(int x, int y); //순환호출로, 모든 좌표를 칠하는 함수 원형 정의

int main() {

print\_screen(); //수행 전에 기본 좌표를 설정해준다.

flood\_fill(0, 0); //x 좌표 y 좌표가 흰색인 임의의 곳을 한군데 설정해준다.

print\_screen(); //잘 수행됬나 출력을 해본다.

return 0;

}

int screen\_read(int x, int y) { //x 좌표 y 좌표에 있는 픽셀 값을 보는 함수 선언

return screen[x][y]; //screen 에 x 좌표 y 좌표를 읽고 가진 값을 반환한다.

}

void write\_pixel(int x, int y) { //x 좌표 y 좌표에 검정색을 칠하는 함수

screen[x][y] = BLACK; //노란색은 칠하지 않으므로 검정색만 칠한다.

//그렇기떄문에 color parameter 를 생성할 필요가 없다.

}

void print\_screen() { //스크린을 출력하는 함수

for (int i = 0; i < WIDTH; i++) { //i 는 width 만큼

for (int j = 0; j < HEIGHT; j++) { //j 는 height 만큼 반복한다.

printf("%d ", screen[i][j]); //screnn 의 i j 좌표를 출력하고 띄운다.

}

printf("\n"); //한줄이 출력됬으면 개행한다.

}

printf("\n\n\n\n\n");

}

void flood\_fill(int x, int y) {

if (screen\_read(x, y) == WHITE) { //x 좌표 y 좌표가 흰색일때만 실행한다.

//\*\*\*\*수행할수록 흰색이 아닌 면적이 줄어든다.\*\*\*\*\*\*

write\_pixel(x, y); //x y 들어온 좌표를 흰색으로 칠해준다.

if (x != WIDTH - 1) { //x 좌표가 배열 인덱스의 최대값이 아니면

flood\_fill(x + 1, y); //x 좌표의 오른쪽도 칠해야 하므로 x + 1 좌표를 확인한다

}

if (x != 0) { //x 좌표가 배열 인덱스의 최소값이 아니면

flood\_fill(x - 1, y); //x좌표의 왼쪽도 칠해야 하므로 x - 1 좌표를 확인한다.

}

if (y != HEIGHT - 1) { //y 좌표가 배열 인덱스의 최대값이 아니면

flood\_fill(x, y + 1); //y 좌표의 오른쪽도 칠해야 하므로 y + 1 좌표를 확인한다.

}

if (y != 0) { //y 좌표가 배열 인덱스의 최소값이 아니면

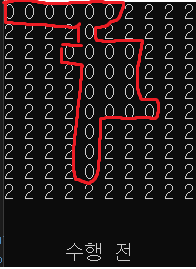
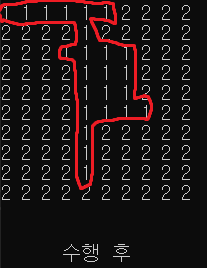
flood\_fill(x, y - 1); //y 좌표의 왼쪽도 칠해야 하므로 y - 1좌표를 확인한다.

}

}

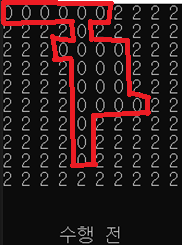
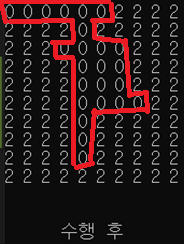
}

1. 실행결과
   1. 흰색이 경계면에 있는 경우와 시작하는 x y 좌표 (0 , 0)가 흰색인 경우

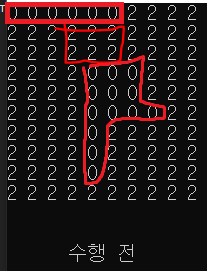
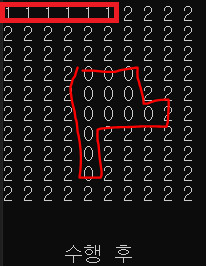
프로그램이 오류 없이 잘 돌고, 연속된 0 모두 1로 바뀌었다.

* 1. 흰색이 경계면에 있는 경우와 시작하는 x y 좌표 (0, 9) 가 노란색인 경우

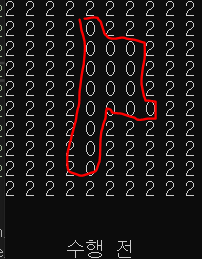
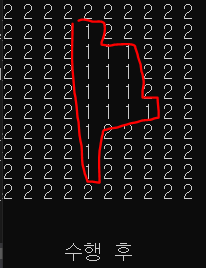
시작좌표 자체가 노란색이므로 if (screen\_read(x, y) == WHITE) 문을 실행하지 못해 원본 배열이 바뀌지 않는다.

* 1. 흰색이 경계면에 있는 경우와 시작하는 x y 좌표(0 , 0)가 흰색이고, 다른 흰색이 연속되지 않는 경우

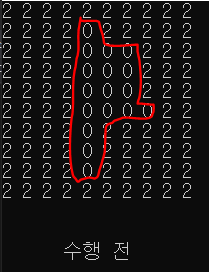
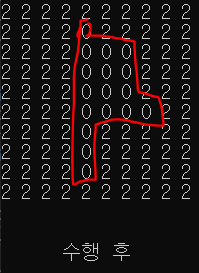
연속됬던 0들은 정상적으로 1로 바뀌지만 연속되지 않은 0 은 그대로 0 임을 볼수 있다 ( 수행 후 사진의 빨간색 )

* 1. 흰색이 경계면에 없는 경우와 시작하는 x y 좌표(1, 4) 가 흰색인 경우

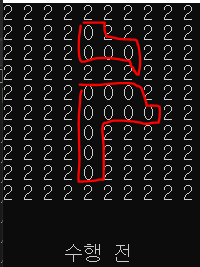
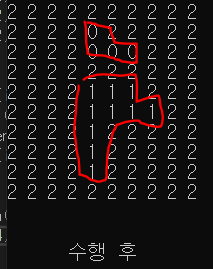
이것도 A 와 마찬가지로 연속된 0 들은 모두 빨간색으로 칠해진다.

* 1. 흰색이 경계면에 없는 경우와 시작하는 x y 좌표가 ( 1 , 3 ) 노란색인 경우

B 와 마찬가지로 시작좌표 자체가 노란색이므로 if (screen\_read(x, y) == WHITE) 문을 실행하지 못해 원본 배열이 바뀌지 않는다.

* 1. 흰색이 경계면에 없는 경우와 시작하는 x y 좌표가 ( 4 , 4 ) 흰색이고, 다른 흰색이 연속되지 않는 경우

연속된 0 들은 칠해지지만, 끊겨진 0 은 1로 바뀌지 않음을 볼 수 있다.