Assignment #1

201910940

정재훈

**1. 소스코드**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct \_node

{

    int height;

    int width;

    int degree;

    struct \_node \*next;

}node;

node \*head, \*tail;

// [ny][nx]

int input\_map[15][15]={

    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0},

    {0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0},

    {0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0},

    {0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0},

    {0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0},

    {0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0},

    {0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0},

    {0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0},

    {0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0},

    {0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0},

    {0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0},

    {0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0},

    {0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0},

    {0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0},

    {0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0}

};

// delta

int dx[8] = {0, 1, 1, 1, 0, -1, -1, -1};

int dy[8] = {-1, -1, 0, 1, 1, 1, 0, -1};

void init\_stack()

{

    head = (node \*)calloc(1, sizeof(node));

    tail = (node \*)calloc(1, sizeof(node));

    head->next = tail;

    tail->next = tail;

}

int push(int x, int y, int d)

{

    node \*t;

    if((t = (node \*)malloc(sizeof(node))) == NULL)

    {

        printf("Out of memory !\n");

        return -1;

    }

    t->height = y;

    t->width = x;

    t->degree = d;

    t->next = head->next;

    head->next = t;

    return 0;

}

node \*pop()

{

    node \*t;

    if(head->next == tail)

    {

        printf("Stack underflow !\n");

        return NULL;

    }

    t = head->next;

    head->next = t->next;

    //free(t);

    return t;

}

void CCA()

{

    node \*pos;

    int i, j, k, nx, ny;

    int label = 2;

    int flag = 0;

    for (i = 0; i < 15; i++){

        for (j = 0; j < 15; j++){

            int x = j;

            int y = i;

            while(1){

                if(flag == 0){

                    int found = 0;

                    for(k = 0; k < 8; k++){

                        nx = x + dx[k];

                        ny = y + dy[k];

                        if (ny < 0 || ny >= 15 || nx < 0 || nx >= 15){

                            continue;

                        }

                        if (input\_map[ny][nx]==1){

                            push(x, y, k);

                            input\_map[ny][nx] = label;

                            x = nx;

                            y = ny;

                            found = 1;

                            break;

                        }

                    }

                    if(head->next==tail){

                        break;

                    }

                    else if(!found){

                        flag = 1;

                    }

                }

                else if(flag == 1){

                    if(head->next==tail){

                        flag = 0;

                        label++;

                        break;

                    }

                    pos = pop();

                    int d = pos->degree;

                    int px = pos->width;

                    int py = pos->height;

                    free(pos);

                    int found = 0;

                    for(k = 0; k < 8; k++){

                        nx = px + dx[(d + k) % 8];

                        ny = py + dy[(d + k) % 8];

                        if (ny < 0 || ny >= 15 || nx < 0 || nx >= 15){

                            continue;

                        }

                        if (input\_map[ny][nx]==1){

                            push(px, py, (d + k) % 8);

                            input\_map[ny][nx] = label;

                            x = nx;

                            y = ny;

                            found = 1;

                            break;

                        }

                    }

                   if(found){

                        flag = 0;

                    }

                }

            }

        }

    }

}

void printMap()

{

    int i, j;

    for (i = 0; i < 15; i++)

    {

        for (j = 0; j < 15; j++)

        {

            printf("%d ", input\_map[i][j]);

        }

        printf("\n");

    }

}

void main()

{

    init\_stack();

    CCA();

    printMap();

}

**2. 실행 결과**

**패턴, 디자인, 패브릭, 예술이(가) 표시된 사진

자동 생성된 설명**

**3. 분석**

교수님께서 수업시간에 설명해주신 방법을 토대로 과제를 진행하였습니다.

우선 stack을 사용하는데 stack에 저장되는 값이 1개가 아닌 3개이므로, 기존의 pop함수가 아닌 node를 반환하도록 하였습니다. 따라서 node \*pos를 선언함과 동시에 pos = pop()을 사용하면 바로 값을 빼고 free(pos)를 통해 메모리를 해제하였습니다.

두 번째로는 flag와 found를 사용하여 기준점을 정하였습니다. Flag의 역할은 무한루프를 돌며, push를 진행할 것인 지, 더 이상 push할 것이 없어 pop을 진행할 것이 지 구분해주는 역할을 합니다. Found의 역할은 flag값이 바뀌기 전에 이웃 값이 있는지 없는지 구별해주는 역할을 합니다.