IOT301 – LẬP TRÌNH C CƠ BẢN

THUYẾT MINH ASSIGNMENT 3
BÀI TOÁN TỌA ĐỘ ĐIỂM

Tên học viên:	Lương Văn Mạnh
Mã học viên:	FX19518
Ngày báo cáo:	06/02/2024

Lập trình C cơ bản
Assignment 3

Thái Nguyên, 02/2024

MỤC LỤC

DANH MỤC HÌNH VẼ, BẢNG BIỂU

NỘI DUNG THUYẾT MINH

1. Danh sách thư viện

STT	Thư viện	Mô tả
1	stdio.h	Thư viện có các hàm nhập xuất + printf: In dữ liệu ra màn hình + scanf: Nhập liệu
2	stdlib.h	Thư viện tiêu chuẩn với các hàm + atoi, atof: chuyển từ string sang giá trị integer/float + malloc, free: cấp phát bộ nhớ động
3	stdbool.h	Thư viện kiểu dữ liệu luận lý Boolean + bool với hai giá trị true, false.

2. Danh sách các biến toàn cục và các macro

STT	Biến toàn cục và macro	Mô tả
1	MAX 10	Giá trị tối đa các phần tử trong hàng đợi queue
2	MAX_ROW 9	Số dòng tối đa của ma trận
3	MAX_COLUMN 9	Số cột tối đa của ma trận

4	typedef struct POINT {
	int row; //tọa độ x của nút
	int column; //tọa độ y của nút
	int value; //giá trị 0 hoặc 1 của nút
	bool visited; //đánh dấu nút đã đi qua
	struct POINT* prev; //con trỏ trỏ đến nút liền trước đó trước khi đi đến nút này.
	} point_t;
		Định nghĩa biến cấu trúc point_t với các thông tin về các nút.

5	matrix[MAX_ROW][MAX_COLUMN];	Ma trận lưu trữ thông của các nút có thể đi qua.

6	typedef struct {
	point_t data[MAX]; //lưu trữ các nút
	int front; //chỉ số đầu hàng đợi
	int rear; //chỉ số cuối hàng đợi
	} Queue;
		Cấu trúc hàng đợi, đưa các nút cần kiểm tra vào để kiểm tra lần lượt theo thứ tự first in first out.

3. Danh sách các hàm

STT	Danh sách hàm	Mô tả
1	int main()	Hàm chính của chương trình
2	bool checkCoordinate(int dong, int cot);	Kiểm tra tọa độ của nút có nằm trong ma trận hay không
3	void initQueue(Queue *q)	Khởi tạo hàng đợi
4	int isEmpty(Queue q);	Kiểm tra hàng đợi rỗng hay không
5	int isFull(Queue q);	Kiểm tra hàng đợi đã đầy hay chưa
6	void enQueue(Queue *q, point_t x);	Đưa nút cần kiểm tra vào hàng đợi
7	point_t deQueue(Queue *q);	Lấy nút cần kiểm tra ra khỏi hàng đợi
8	void findSurroundingPoint(int dong, int cot, point_t surroundingPnt[4], int* count);	Tìm các nút có thể đi tới từ nút hiện tại.
9	void findShortestPath(int dong, int cot);	Tìm đường ngắn nhất từ O(0,0) đến A(dòng, cột)

4. Hàm bool checkCoordinate(int dong, int cot)

a. Mô tả

Hàm checkCoordinate() thực hiện việc kiểm tra tọa độ dòng = x, cột = y với điều kiện 0 <= x < MAX_ROW và 0 <= y < MAX_COLUMN.

b. Tham số truyền vào

Tọa độ dòng cột x, y.

Trong đó x, y là số nguyên(int)

c. Giá trị trả về

Kiểu dữ liệu trả về là bool. Trong đó

+ true : nút nằm trong ma trận.

+ false: nút nằm ngoài ma trận.

5. Hàm void initQueue(Queue *q)

a. Mô tả

Hàm initQueue() thực hiện thiết lập chỉ số đầu hàng đợi.

b. Tham số truyền vào

Hàng đợi Queue *q.

Trong đó q là pointer đến hàng đợi có cấu trúc point_t(kiểu struct POINT)

c. Giá trị trả về

Hàm void nên không có giá trị trả về.

6. Hàm int isEmpty(Queue q)

a. Mô tả

Hàm isEmpty() thực hiện so sánh chỉ số đầu front và chỉ số cuối rear của hàng đợi.

b. Tham số truyền vào

Hàng đợi Queue q.

Trong đó q là hàng đợi có cấu trúc point_t(kiểu struct POINT)

c. Giá trị trả về

Toán tử so sánh " < " trả về giá trị 0 và 1 tương đương với false và true.

7. Hàm int isFull(Queue q)

a. Mô tả

Hàm isFull() thực hiện so sánh chỉ số cuối rear của hàng đợi và số lượng phần tử tối đa mà hàng đợi có thể chứa, xem hàng đợi đã đầy hay chưa.

b. Tham số truyền vào

Hàng đợi Queue q.

Trong đó q là hàng đợi có cấu trúc point_t(kiểu struct POINT)

c. Giá trị trả về

Nếu kết quả so sánh bằng nhau trả về giá trị 1, khác nhau trả về giá trị 0 tương đương với true và false.

8. Hàm void enQueue(Queue *q, point_t x)

a. Mô tả

Hàm enQueue() thực hiện kiểm tra xem hàng đợi đã đầy hay chưa, nếu chưa thì thêm phần tử vào cuối hàng đợi có thể chứa, thay đổi chỉ số cuối (rear) hàng đợi.

b. Tham số truyền vào

Hàng đợi Queue q.

Cấu trúc point_t x để lưu thông tin phần tử thêm vào.

Trong đó q là hàng đợi có cấu trúc point_t(kiểu struct POINT), x là cấu trúc (kiểu struct POINT).

c. Giá trị trả về

Hàm void nên không có giá trị trả về.

9. Hàm point_t deQueue(Queue *q)

a. Mô tả

Hàm deQueue() thực hiện kiểm tra xem hàng có đã rỗng hay không, nếu chưa thì lấy ra phần tử đầu của hàng đợi, lưu vào cấu trúc point_t d, thay đổi chỉ số đầu (front) hàng đợi.

b. Tham số truyền vào

Pointer trỏ đến hàng đợi Queue q.

Cấu trúc point_t d để lưu thông tin phần tử lấy ra.

Trong đó q là hàng đợi có cấu trúc point_t(kiểu struct POINT), d là cấu trúc (kiểu struct POINT).

c. Giá trị trả về

Trả về biến có cấu trúc point_t là phần tử được lấy ra.

10. Hàm void findSurroundingPoint(int dong, int cot, point_t surroundingPnt[4], int* count)

a. Mô tả

Hàm findSurroundingPoint() thực hiện kiểm tra xem 4 nút xung quanh (trên, dưới, phải, trái) xem nút nào thỏa điều kiện để đi đến được.

b. Tham số truyền vào

Hàm sử dụng 4 tham số.

dong, cot kiểu int lưu giá trị dòng, cột.

Lưu các nút thỏa mãn vào mảng cấu trúc surroundingPnt[], lưu số lượng nút vào count.

Trong đó surroundingPnt là mảng cấu trúc kiểu struct POINT.

c. Giá trị trả về

Hàm void nên không có giá trị trả về.

11. Hàm void findShortestPath(int dong, int cot)

a. Mô tả

Hàm findShortestPath() thực hiện kiểm tra các nút trong ma trận matrix[dong][cot] bắt đầu từ matrix[0][0] đến matrix[dong][cot], sử dụng findSurroundingPoint() để tìm các nút có thể đi tới rồi đưa vào hàng đợi Queue queue để kiểm tra lần lượt theo FIFO. Đánh dấu các nút đã đi qua là visited để không kiểm tra lại. Dùng pointer prev để lưu địa chỉ của nút trước đó. Lưu các nút đã đi qua này vào ma trận matrix. Sau khi kết thúc sẽ tạo nên các đường đi (ví dụ ở đây là 3 đường linked list), kiểm tra đến khi nút matrix[dong][cot] được tìm thấy, lúc này con trỏ sẽ ở vị trí matrix[dong][cot]. Ta sẽ dùng một mảng cấu trúc point_t path[] để lưu các nút này lần lượt từ matrix[dong][cot] đến matrix[0][0]. Sau đó, in mảng này ra màn hình theo thứ tự từ cuối lên đầu thì ta sẽ có đường đi ngắn nhất.

b. Tham số truyền vào

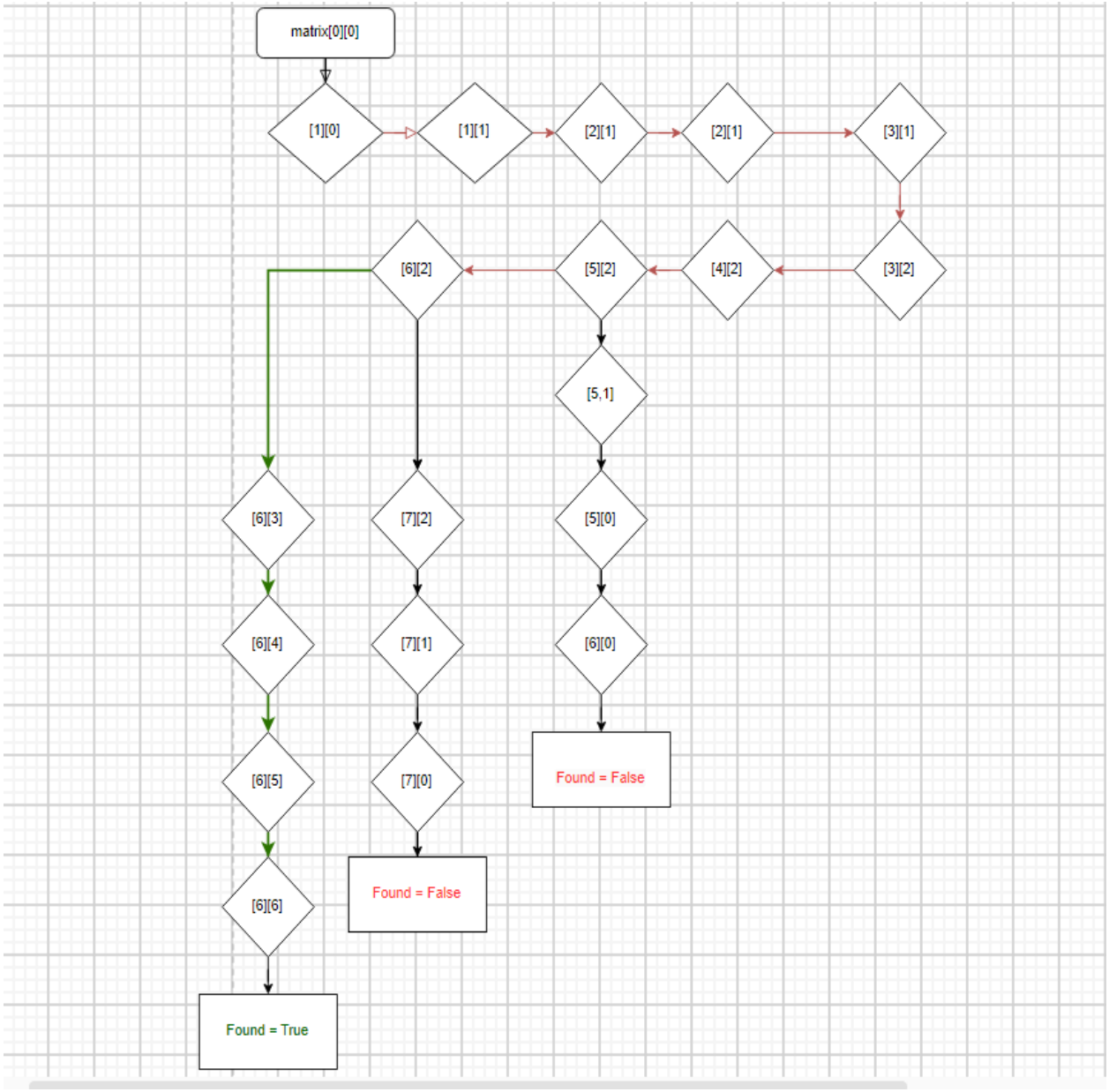
Hàm sử dụng 2 tham số dong, cot kiểu int lưu giá trị dòng, cột.

c. Giá trị trả về

Hàm void nên không có giá trị trả về.

12. Luồng xử lý code

(Nếu vẽ được sơ đồ thuật toán sẽ được đánh giá cao hơn)



Các bước thực xử lý:
Bước 1: Tạo một mảng lưu giá trị các nút trong ma trận, nhập giá trị vào ma trận matrix rồi hiển thị;
Bước 2: Nhập tọa độ dòng cột của ô đích đến A(dong, cot), kiểm tra xem tọa độ có thỏa mãn yêu cầu.
Bước 3: Gọi hàm findShortestPath() để tìm và hiển thị các nút từ O(0,0) đến A(dong, cot).