|  |  |
| --- | --- |
| Học trực tuyến CNTT, học lập trình từ cơ bản đến nâng cao |  |

**IOT301 – LẬP TRÌNH C CƠ BẢN**

**BÀI TOÁN TỌA ĐỘ ĐIỂM**

**THUYẾT MINH ASSIGNMENT 3**

|  |  |
| --- | --- |
| Tên học viên:  Mã học viên:  Ngày báo cáo: | Nguyễn Quang Huy  FX09149  22/10/2022 |

**NỘI DUNG THUYẾT MINH**

**1.Các câu lệnh tiền xử lý:**

**1.1#include <stdio.h>:trỏ đến thư viện có các hàm scanf,printf..**

**1.2#include<stdlib.h>:trỏ đến thư viện dùng để xúc tiến nhiều phép toán,cấp phát vùng nhớ..**

**1.3#include<stdbool.h>trỏ đến thư viện thực hiện các câu lệnh có giá trị đúng sai..**

**2.Tạo các cấu trúc và biến cục bộ theo mẫu.**

**3.Các hàm trong bài:**

**3.1.Khởi tạo hàm đợi queue : void initQueue(Queue\*q):**

B1:Khởi tạo hệ số đầu của hàng đợi q->front=0.

B2:Khởi tạo hệ số cuối của hàng đợi q->rear=-1.

**3.2.Kiểm tra hàm rỗng : int isEmpty(Queue q):**

Khi front > rear thì hàm rỗng.

**3.3.Kiểm tra hàm đầy :int isFull(Queue q):**

Nếu rear = MAX-1 thì hàm đã đầy.

**3.4 Cho thêm một phần tử vào hàm : void enQueue(Queue\*q,point\_t x):**

Xử dụng câu điều kiện if kiểm tra hàm đã đầy chưa :

\*Nếu hàng chưa đầy cho thêm một phần tử vào cuối hàng

\*Nếu hàng đã đầy thì bỏ qua những phần tử mà front đã đi qua

**3.5 Lấy phần tử đầu tiên ra khỏi hàng point\_t deQueue(Queue \*q):**

B1: Nếu hàm không rỗng tăng front lên 1 và chuyển dữ liệu cho biến d

B2 :Nếu hàm rỗng đặt lại front và rear.

**3.6 Hàm kiểm tra điểm bất kì có tọa độ nằm trong mảng hay không: bool checkCoordinate(int dong,int cot):**

\*Nếu 0<=dong<9 và 0<=cot<9 thì dong và cot nằm trong mảng.

**3.7 Hàm tìm những điểm (dong,cot )có giá trị bằng 1 và lưu vào mảng surroundingPnt : int findSurroundingpoint(int dong,int cot, point\_t surroundingPnt[]):**

\*B1.Kiểm tra điểm nằm bên phải có trong ma trận và có giá trị =1

\*B1.Kiểm tra điểm nằm bên dưới có trong ma trận và có giá trị =1

\*B1.Kiểm tra điểm nằm bên trái có trong ma trận và có giá trị =1

\*B1.Kiểm tra điểm nằm bên trên có trong ma trận và có giá trị =1

**3.8 Hàm tìm đườngngắn nhất từ 0(0,0) đến A(dong,cot): void findShortestPath (int dong, int cot):**

B1: khai báo một hàng đợi Queue.

B2:Khai báo hàm init Queue() để khởi tạo hàm đợi đã được khai báo

B3: Đánh dấu nút matrix[0][0] là đã đi qua

B4: xử dụng hàm enQueue();để cho nút matrix[0][0] vào hàng đợi.

B5:Gọi hàm deQueue(q) để lấy phần tử đầu của hàm để lưu vào biến temp

B6: Gọi hàm surroundingPnt() để tìm các nút xung quanh pp[] của nút p vừa lấy ra khỏi hàng đợi

B7: Dùng vòng for duyệt lần lượt các nút chung quanh vừa được tìm thấy ở trên

B8: Đánh dấu nút pp[i] là đã đi qua (pp[i].visited = true)

B9:Lưu địa chỉ của nút trước đó vào

B10: Nếu nút có cùng tọa độ với nút cần tìm thì thoát khỏi vòng lặp

B11: cho nút pp[i] vào hàng đợi bằng hàm enQueue()

B12:Hiển thị các nút xung trên đường đi của nút được xét.

B13:Khi tìm thấy nút cần tìm,gán con trỏ m cho địa chỉ của mảng &matrix[dong][cot] và khai báo một biến sum để tính số lượng nút trên đường đi ngắn nhất,khởi tạo mảng point\_t mang[100] để lưu trữ tạm thời các nút trên đường đi.

B14: xử dụng vòng lặp While để tạo danh sách liên kết các nút và vòng lặp for để in ngược lần lượt các nút là đường đi ngắn nhất

B15:else sẽ không có đường đi từ 0(0,0) đến A(dong,cot).

**B.9 Hàm main**

B1: Lưu ma trận đã cho vào mảng hai chiều sử dụng vòng lặp for

B2:Sử dụng vòng lặp while để nhập tạo độ nếu tạo độ đã cho thỏa mãn điều kiện thì thoát vòng lặp nếu không thỏa mãn thì nhập lại