**ASSIGNMENT 3**

**Môn học:** IOT301x Lập trình C cơ bản

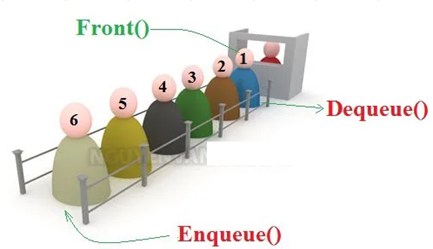
**Họ và tên:** Nguyễn Ngọc Thiện

**Mã học viên:** FX-10909

1. **Danh sách các hàm**

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên hàm** | **Chức năng** |
| **1.** | void initQueue(*Queue* \**q*) | Khởi tạo queue rỗng |
| **2.** | int isEmpty(*Queue* *q*) | Kiểm tra queue có rỗng hay không |
| **3.** | int isFull(*Queue* *q*) | Kiểm tra queue có đầy hay không |
| **4.** | void enQueue(*Queue* \**q*, *point\_t* *x*) | Thêm x vào đầu queue |
| **5.** | *point\_t* deQueue(*Queue* \**q*) | Lấy phần tử đầu ra khỏi queue |
| **6.** | bool checkCoordinate(int *x*, int *y*) | Kiểm tra toạ độ (x, y) có hợp lệ không |
| **7.** | void findSurroundingPoint(int *x*, int *y*, int \**count*) | Tìm các điểm xung quanh toạ độ (x, y). Count chứa số lượng các điểm xung quanh hợp lệ |
| **8.** | void findShortestPath(int *x*, int *y*) | Tìm đường đi ngắn nhất từ toạ độ (0, 0) đến toạ độ (x, y) |
| **9.** | int main() | Hàm thực thi chính |

1. **Hàng đợi**

****

* **Hàng đợi** được tổ chức như một mảng một chiều, để quản lý hàng đợi, chúng ta dùng những thuộc tính và các hàm sau:
  + **Front:** thuộc tính này chỉ ra vị trí đầu hàng đợi
  + **Rear:** thuộc tính này chỉ ra vị trí cuối hàng đợi
  + **Dequeue():** lấy một phần tử ra khỏi hàng đợi
  + **Enqueue():** đưa một phần tử vào hàng đợi
  + **isEmpty():** kiểm tra hàng đợi rỗng hay không
  + **isFull():** kiểm tra hàng đợi có đầy hay không

1. **Chi tiết các hàm**
   1. **Hàm checkCoordinate(int x, int y)**

* Kiểm tra nếu toạ độ bé hơn 0 hoặc lớn hơn kích thước hàng hoặc cột tối đa thì trả về **false**. Ngược lại trả về **true**.
  1. **Hàm findSurroundPoint(int x, int y, int \*count)**
* Kiểm tra các toạ độ bên trái, bên phải, bên dưới, bên trên bằng hàm **checkCoordinate(x, y)**. Nếu các toạ độ này hợp lệ, thì cập nhật vào mảng **surroundingPnt** và tăng biến **count**.
  1. **Hàm findShortestPath(int x, int y)**
* **Bước 1:** Khởi tạo queue và đưa phần tử đầu tiên vào queue
* **Bước 2:** Lấy phần tử ra khỏi queue và kiểm tra xung quanh.
* **Bước 3:** Đánh dấu toạ độ hiện tại là đã đi qua. Cập nhật địa chỉ của toạ độ trước đó. Nếu toạ độ hiện tại là đích đến thì chuyển tới **Bước 5**.
* **Bước 4:** Đưa phần tử tiếp theo vào queue
* **Bước 5:** In toạ độ ra màn hình và kết thúc
  1. **Hàm int main()**
* **Bước 1:** Khởi tạo Matrix
* **Bước 2:** Nhập toạ độ (x, y) mong muốn
* **Bước 3:** Thực thi hàm **findShortestPath(x, y).**
  1. **Struct POINT**
* **Các thuộc tính của cấu trúc POINT:**
  + **int row** toạ độ x của nút
  + **int col** toạ độ y của nút
  + **int value**  giá trị 0 / 1 của nút
  + **bool visited**  đánh dấu nút đã được đi qua là true/ false
  + **struck *POINT* \*prev** con trỏ trỏ đến nút trước đó trước khi đi đến nút này
  + **point\_t** gán biến point\_t cho struct