Programming Practice: Queue

1. A queue is a container with first-in-first-out elements. A queue can be viewed as a ticket line that the first person joining the line is called the head of the line and the last person joining the line is called the tail of the line. Hence, an enqueue operation is to add a new element to the tail of a queue and a dequeue operation is to remove an element from the head of a queue.

   Write a C project to define a data type representing queues using array of integers and define the following queue operations:

```
#define max 100 // Maximum 100 element.
// Type definition of queues using an array.
// The queue elements of integers.
typedef struct {
  int elem[max]; // Holding elements of a queue.
  int head; // Index of the queue head.
  int tail; // Index of the queue tail.
  int cnt; // Element count in a queue.
} Queue;
```

   1. **void** initial_queue(Queue *): Set a queue to empty, i.e., reset head and tail of a queue.
   2. **void** enqueue(Queue *, **int**): Add an element to the tail of a queue.
   3. **int** dequeue(Queue *): Remove an element from the head of a queue.
   4. **int** head(Queue): Get the element at the head of a queue.
   5. **int** is_empty(Queue): Check if a queue is empty or not.
   6. **void** print_queue(Queue): Print elements of a queue from the head to the tail.

The main program is a loop to repeatedly read an integer command between 1 and 6: (1) Enqueue, (2) Dequeue, (3) Head element, (4) Empty test, (5) Print queue, and (6) Quit.

Solutions: queue_array_proj.dev, queue_array.h, queue_array.c, and queue_array_main.c. Example of program execution: