자료구조 실습과제

원형 큐 프로그램

5671144 강범창

1.1 원형 큐 프로그램에서 사용할 typedef 들을 선언합니다.

Element 즉 큐에 들어가는 요소들은 int 형을 가집니다. 그리고 QueueType 의 경우 큐를 구현하기 위해 사용하며, data는 요소들을 저장해 놓기 위한 element 배열이고 front 는

```
typedef int element;
typedef struct
{
    int front;
    int rear;
    element data[MAX_QUEUE_SIZE];
} QueueType;
```

1.2 메인에서 사용할 변수들을 초기화 합니다.

```
QueueType queue; // 원형 큐를 사용할 때 사용합니다.
int menu;   // menu 입력을 받기위해 사용합니다.
int item; // 큐에 넣을 수를 넣거나 받기 위해 사용합니다.
init(&queue); // 큐를 초기화합니다.
```

1.3 원형 큐를 구현합니다.

Init 함수입니다. Queue를 초기화하는데 사용합니다.

```
void init(QueueType *queue)
{
    queue->front = 0;
    queue->rear = 0;
```

```
}
```

isEmpty 함수 입니다. queue의 앞부분과 뒷부분이 같다면 큐에 값이 없다는 뜻이므로 비었다는 의미로 1을 return 하고 아니라면 0을 return을합니다.

```
int isEmpty(QueueType *queue)
{
    return queue->front == queue->rear ? 1 : 0;
}
```

isFull 함수입니다. Queue 의 rear즉 queue의 뒷부분을 MAX_QUUE size으로 나머지를 구해 front 앞부분과 같다면 queue가 꽉 찼다 생각하고 1을 return 합니다. 아니라면 0을 return 합니다. MAX_QUEUE_SIZE으로 나머지를 가지고 하는 이유는 원형 큐를 구현하기 위해서 이며, 이렇게 구현을 함으로 1 , 2 , 3 , 4 , 0 이런 식으로 원형으로 반복해서 돌아갈 수 있기 때문입니다.

```
int isFull(QueueType *queue)
{
    return (queue->rear + 1) % MAX_QUEUE_SIZE == queue->front ? 1 : 0;
}
```

Enqueue 함수입니다. Queue에 값을 넣는 함수이며, 위에서 설명하듯 원형 큐에 값을 넣습니다. 이 때 rear(끝부분) 의 값을 변경시켜주고 item 을 data 끝부분에 넣어줍니다. 이때 full 인지 확인을 안 하는 이유는 main 에서 실행 전 꽉 찼는지 확인을 하기 때문에 따로 해주지 않습니다.

```
void enqueue(QueueType *queue, element item)
{
    // 원형이기에 (queue->rear + 1) % MAX_QUEUE_SIZE
    queue->rear = (queue->rear + 1) % MAX_QUEUE_SIZE;
    queue->data[queue->rear] = item;
}
```

Dequeue 함수입니다. 이것은 원형 큐에서 값을 가져오기 위해 사용하며 큐가 비어 있을 경우에는 비어 있다는 메시지를 프린트하고 return -1 을 함으로 비어 있다는 것을 알립니다. 아닌 경우 front의 값을 변경하고 front위치의 값을 return 합니다. 그리고 front의 값은 원형 큐의 방식으로 1 2 3 4 0 이런 식으로 정해집니다.

```
element dequeue(QueueType *queue)
{
	if (isEmpty(queue))
	{
		printf("Queue is empty₩n");
		return -1;
	}
	queue->front = (queue->front + 1) % MAX_QUEUE_SIZE;
	return queue->data[queue->front];
}
```

printQueue 큐를 프린트하는 함수입니다. Queue의 front(앞부분) rear(끝부분) 위치의 값을 출력하고 만약 queue 가 비어 있지 않다면 queue를 출력합니다.

```
void printQueue(QueueType *queue)
{
	printf("Queue(front = %d rear = %d): ", queue->front, queue->rear);
	if (!isEmpty(queue))
	{
		int i = queue->front;
		do
		{
			i = (i + 1) % MAX_QUEUE_SIZE;
			printf("%d ", queue->data[i]);
			if (i == queue->rear)
				break;
		} while (i != queue->front);
	}
	printf("₩n");
```

```
}
```

1.3 메인에서 menu를 처리하는 부분을 작성합니다.

Menu를 출력하며 1번 메뉴를 선택 시 값을 큐에 넣는 함수를 실행시키고, 2번의 경우 값을 빼는 함수를 실행합니다. 이때 값이 꽉 차 있다면 차 있다는 메시지를 출력하고 메뉴실행을 종료합니다. 3번의 경우 큐를 프린트하는 함수를 실행합니다. 마찬가지로 비어 있다면 그 메시지를 출력하고 메뉴를 종료합니다. 4번을 입력하면 프로그램을 종료한다는 메시지를 출력하고 프로그램을 종료합니다.

```
while (1)
    {
        printf("₩n");
        printf("==== Menu ====₩n");
        printf("1. Input data and Enqueue₩n");
        printf("2. Dequeue and Print Data₩n");
        printf("3. Print Queue₩n");
        printf("4. Exit₩n");
        printf("Select number: ");
        scanf("%d", &menu);
        printf("₩n");
        switch (menu)
        {
        case 1:
            if (isFull(&queue))
            {
                printf("Queue is Full₩n");
                break;
            }
            printf("Input the data : ");
            scanf("%d", &inputnum);
            enqueue(&queue, inputnum);
            printf("Enqueue: %d₩n", inputnum);
```

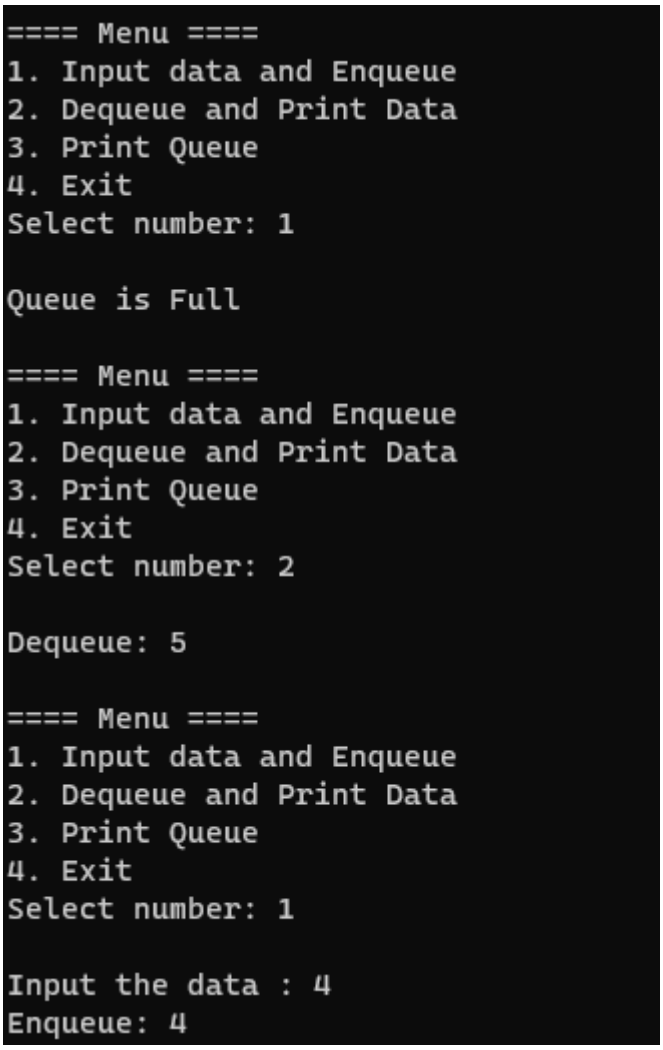
```
            break;
        case 2:
            item = dequeue(&queue);
            if (item != -1)
            {
                printf("Dequeue: %d₩n", item);
            }
            break;
        case 3:
            printQueue(&queue);
            break;
        case 4:
            printf("Exit the program.₩n");
            exit(1);
            break;
        default:
            break;
        }
    }
```

아래는 실행 결과입니다.

값이 꽉 찼는지 확인을 하고 찾으면 꽉 찼다고 출력하는 예시입니다. 다시 시도를 해도 찼다는 메시지를 출력합니다.

```
==== Menu ====
1. Input data and Enqueue
2. Dequeue and Print Data
3. Print Queue
4. Exit
Select number: 1

Input the data : 9
Enqueue: 9

==== Menu ====
1. Input data and Enqueue
2. Dequeue and Print Data
3. Print Queue
4. Exit
Select number: 3

Queue(front = 0 rear = 4): 5 6 7 9

==== Menu ====
1. Input data and Enqueue
2. Dequeue and Print Data
3. Print Queue
4. Exit
Select number: 1

Queue is Full

==== Menu ====
1. Input data and Enqueue
2. Dequeue and Print Data
3. Print Queue
4. Exit
Select number: 1

Queue is Full
```

아래는 값을 뺀 다음 추가를 할 시 정상적으로 값이 들어가는 모습을 볼 수 있습니다.



이후 값들을 모두 뺀 다음 empty를 출력하는 예시입니다. 이것도 다시시도 해도 똑같이 empty 를 출력합니다.

```
Input the data : 4
Enqueue: 4

==== Menu ====
1. Input data and Enqueue
2. Dequeue and Print Data
3. Print Queue
4. Exit
Select number: 2

Dequeue: 6

==== Menu ====
1. Input data and Enqueue
2. Dequeue and Print Data
3. Print Queue
4. Exit
Select number: 2

Dequeue: 7

==== Menu ====
1. Input data and Enqueue
2. Dequeue and Print Data
3. Print Queue
4. Exit
Select number: 2

Dequeue: 9

==== Menu ====
1. Input data and Enqueue
2. Dequeue and Print Data
3. Print Queue
4. Exit
Select number: 2

Dequeue: 4

==== Menu ====
1. Input data and Enqueue
2. Dequeue and Print Data
3. Print Queue
4. Exit
Select number: 2

Queue is empty

==== Menu ====
1. Input data and Enqueue
2. Dequeue and Print Data
3. Print Queue
4. Exit
Select number: 2

Queue is empty
```