자료구조론

Queue

```
u@hataeseong-ui-MacBook-Pro:~/Desktop/
2017_CSE2010_2016025041/HW3$./a.out input1.txt
init front = 0, rear = 0
enqueue() = 0
front = 0, rear = 1
enqueue() = 1
front = 0, rear = 2
enqueue() = 2
front = 0, rear = 3
enqueue() = 3
front = 0, rear = 4
Queue is full
```

- 실행결과 : 일치

```
char command;
FILE *input;
element element_;
QueueType *queue_ = (QueueType *)malloc(sizeof(QueueType));
input = fopen(argv[1], "r");
init(queue_);
```

- 필요한 변수 선언

```
while(1){
        command = fgetc(input);
        if(feof(input)) break;
        switch(command){
                case 'e':
                        fscanf(input, "%d", &element_);
                        enqueue(queue_, element_);
                        break;
                case 'd':
                        dequeue(queue_);
                        break;
                case 'p':
                        peek(queue_);
                        break;
                default:
                        ;
        }
  }

  fclose(input);
  free(queue_);
```

- 파일을 읽어와서 입력에 따라 각각 실행함수 호출

```
int is_empty(QueueType *q)
{
  if(q->front == q->rear)
        return 1;
  else
        return 0;
}
```

- 비어있으면(front = rear) 이면 True 리턴, 아니면 False 리턴

```
int is_full (QueueType *q)
{
  if(q->rear == q->front - 1)
        return 1;
  else
        return 0;
}
```
- 꽉차있으면 True 리턴, 아니면 False 리턴

```
void enqueue( QueueType *q, element item )
{
  if(is_full(q)){
        char* message = "Queue is full";
        error(message);
  }
  else{
        q->queue[(q->rear + 1) % MAX_QUEUE_SIZE] = item;
        q->rear = (q->rear + 1) % MAX_QUEUE_SIZE;
        printf("enqueue() = %d\nfront = %d, rear = %d\n", item, q->front, q->rear);
  }
}
```
- 큐가 가득차있을경우 에러메시지 리턴, 아닐경우 rear를 증가후 큐에 저장

```
element dequeue(QueueType *q)
{
  if(is_empty(q)){
        char* message = "Queue is empty";
        error(message);
  }
  else{
        int tmp = q->queue[(q->front + 1) % MAX_QUEUE_SIZE];
        q->queue[(q->front + 1) % MAX_QUEUE_SIZE] = 0;
        q->front = (q->front + 1) % MAX_QUEUE_SIZE;
        printf("dequeue() = %d\nfront = %d, rear = %d\n", tmp, q->front, q->rear);
        return tmp;
  }
}
```

- 큐가 비어있으면 에러메시지 리턴, 아닐경우 front를 증가후 front에 저장된 값 삭제

```
element peek(QueueType *q)
{
  if(is_empty(q)){
        char* message = "Queue is empty";
        error(message);
  }
  else{
        printf("peek() = %d\n", q->queue[(q->front + 1) % MAX_QUEUE_SIZE]);
        return q->queue[(q->front + 1) % MAX_QUEUE_SIZE];
  }
}
```

- 큐가 비어있으면 에러메시지 리턴, 아니면 front+1에 저장된 값 리턴