자료구조 실습 보고서

[제 09 주] 배열로 구현된 환형 큐

제출일	2017/05/08
학 번	201000287
소 속	일어일문학과
이 름	유다훈

1 프로그램 설명서

1 주요 알고리즘 및 자료구조

- 알고리즘
  - 입출력
    - ◆ 키보드에서 문자를 반복 입력 받는다.
      - i 한 번에 한 개의 문자를 입력 받는다.
    - ◆ 입력 받은 문자가 '!' 이라면 더이상 입력을 받지 않고 종료한다.
    - ◆ 영문자('A' ~ 'Z' / 'a' ~ 'z')를 입력 받는다면 큐에 삽입한다.
      - i 입력 받은 영문자가 큐에 삽입되면 메세지를 출력한다.
      - ii 만일 큐가 꽉 차 있다면 꽉 차 있다는 메세지를 출력한다.
    - ◆ 숫자문자('0' ~ '9') 를 입력 받으면 해당 수 만큼 큐에서 삭제한다.
      - i 숫자를 입력 받으면 숫자만큼의 원소를 삭제한다는 메세지를 출력한다.
      - ii 삭제될 때 마다 원소가 삭제되었다는 메세지를 출력한다.
      - iii 삭제를 시도하였으나 큐가 비어 있으면 큐에 원소가 없다는 메세지를 출력한다.
    - ◆ '-'(하이픈)을 입력 받으면 큐의 맨 앞 원소를 삭제한다.
      - i 삭제할 때 마다 원소가 삭제되었다는 메세지를 출력한다.
      - ii 삭제를 하려는데 큐가 비어 있으면 큐에 원소가 없다는 메세지를 출력한다.
    - ◆ '/'(슬래시)를 입력 받으면 큐의 내용을 큐의 앞부터 뒤까지 차례로 출력한다.
    - ◆ '^'(곡절부호)를 입력 받으면 큐의 맨 앞의 원소의 값을 출력한다.
      - i 큐의 내용은 변하지 않고 맨 앞 원소만 출력한다.
      - ii 만일 큐가 비어 있다면 큐에 원소가 없다는 메세지를 출력한다.
    - ◆ '#'(샵)을 입력 받으면 큐의 사이즈를 출력한다.
    - ◆ 그 밖의 문자들을 입력 받으면 의미 없는 문자가 입력되었다는 메세지를 출력하고 무시한다.

- 자료구조
  - FIFO 구조의 큐를 구현하는 제네릭 타입의 배열
    - ◆ Queue : First-In-First-Out 형태의 자료구조
  - 각종 입출력 시 알림을 전달하는 Enum
  -

1 함수 설명서

Class	AppController			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	AppController()	없음	없음	_appView 변수와 입력받은 문자와 추가된 문자, 무시된 문자의 수를 초기화하는 메소드를 실행하는 생성자 메소드
	run()	없음	없음	환형 큐 프로그램을 실행하는 실행 메소드
	void showMessage(MessageID aMessageID)	에러 메세지 혹은 알림	없음	에러 메세지 혹은 알림에 따라 메세지를 출력하는 메소드
	void initCharCounts()	없음	없음	입력받은 문자, 추가된 문자, 무시된 문자의

				갯수를 0 으로 초기화하는 메소드
	void countAdded()	없음	없음	추가글자 카운트 상승
	void countIgnored()	없음	없음	무시글자 카운트 상승
	void countInputChar()	없음	없음	입력글자 카운트 상승
	void showFrontElement()	없음	없음	큐 내부의 맨 앞의 원소를 출력지시
	void showQueueSize()	없음	없음	큐 내부의 전체 원소 갯수 출력 지시
	void showAll()	없음	없음	큐 내부의 원소를 앞에서부터 순서대로 출력 지시
	void add(Character anElement)	문자	없음	문자를 큐에 삽입
	void removeOne()	없음	없음	큐 내부 원소 1 개 삭제
	void removeN(int numberOfCharsToBeRemoved)	삭제하고 싶은 원소의 갯수	없음	큐 내부의 원소를 전달받은 수 만큼 삭제
	void conclusion()	없음	없음	큐 내부의 원소를 하나하나 삭제하며 출력지시 및 입력글자, 추가글자, 무시글자 의 갯수를 출력.

Class	AppView			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	AppView()	없음	없음	생성자 메소드 값을 입력받는 스캐너 생성
	String inputString()	없음	문자열값	입력 받은 문자열을 리턴하는 메소드
	char inputCharacter()	없음	문자값	입력받은 문자값의 맨 앞 글자(알파벳 1 개)만 리턴하는 메소드
	void outputMessage(String aMessage)	문자열	없음	전달 받은 문자열을 출력하는 메소드
	void outputAdd(char anElement)	문자값	없음	큐에 삽입된 원소가 무엇인지 메세지를 출력하는 메소드
	void outputElement(char anElement)	문자값	없음	큐의 원소 한 개를 출력하는 메소드
	void outputFrontElement(char anElement)	문자값	없음	큐의 맨 앞 원소를 출력하는 메소드
	void outputQueueSize(int aQueueSize)	정수	없음	큐의 사이즈를 출력하는 메소드
	void outputRemove(char anElement)	문자값	없음	큐에서 삭제된 문자값을 전달받아 무엇이 삭제되었는지 출력하는 메소드
	void outputRemoveN(int numberOfCharsToBeRemoved)	정수	없음	큐에서 전달받은 정수값 만큼의 원소를 삭제하겠다는 알림 메세지를 출력하는 메소드
	void outputResult(int aNumberOfInputChars, int	입력받은 문자갯수, 무시된 문자	없음	입력받은 문자 갯수, 무시된 문자 갯수, 저장된 문자 갯수를 출력하는

	aNumberOfIgnoredChars, int aNumberOfAddedChars)	갯수, 저장된 문자 갯수		메소드

Class	CirculaArrayQueue<T>			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	CircularArrayQueue()	없음	없음	디폴트값으로 큐의 최대 용량을 초기화하는 생성자 메소드
	CircularArrayQueue(int givenCapacity)	큐의 최대 용량값	없음	입력 받은 값으로 큐의 최대 용량과 큐 역할을 할 배열의 최대값을 초기화 하는 생성자 메소드
	boolean isEmpty()	없음	boolean	큐가 비어 있으면 true, 비어있지 않으면 false 를 리턴하는 메소드
	boolean isFull()	없음	boolean	큐가 꽉 차 있다면 true, 그렇지 않다면 false
	int size()	없음	큐 내 원소의 갯수	큐 내부의 원소의 총 갯수 출력
	T elementAT(int aPosition)	원소가 존재하는 위치	원소	입력 받은 위치에 존재하는 스택의 원소를 반환
	T frontElement()	없음	원소	큐 내부의 맨 앞 원소를 리턴
	boolean enQueue(T anElement)	원소	boolean	큐에 전달받은 원소를 입력하면 true, 꽉 차서 입력하지 못하면 false 를 반환하는 메소드
	T deQueue()	없음	원소	삭제할 원소를 반환하는 메소드
	void clear()	없음	없음	큐 내부를 null 값으로 초기화시키는 메소드. 큐를 비우게 하는 것.

2 종합 설명서

- 평범한 영문자를 입력했을 때
  - CircularArrayQueue 의 enQueue()메소드를 이용하여 큐에 넣는다.
  - 만약 꽉 차 있다면 isFull()메소드에 의하여 false 가 반환되고, 큐에 원소가 꽉 찼다는 메세지가 출력된다.
- 숫자문자를 입력했을 때
  - AppView 의 outputRemoveN() 메소드를 통하여 입력받은 숫자만큼의 원소가 삭제된다는 알림 메세지를 출력한다.
  - AppController 의 removeN()메소드를 통해 입력 받은 숫자만큼 반복하며 removeOne() 메소드를 실행한다.
  - removeOne()은 CircularArrayQueue 의 deQueue()메소드를 실행시켜 원소를 삭제하고, 삭제된 원소를 반환하여 AppView 의 outputRemove 를 통해 출력한다.
  - 만일 CircularArrayQueue 의 isEmpty()를 통해 큐가 비어 있다면 에러 메세지를

출력한다.

- '-'(하이픈) 을 입력했을 때
  - removeOne()을 통해 큐의 원소를 하나 삭제한다.
  - removeOne()은 CircularArrayQueue 의 deQueue()메소드를 실행시켜 원소를 삭제하고, 삭제된 원소를 반환하여 AppView 의 outputRemove 를 통해 출력한다.
  - 만일 CircularArrayQueue 의 isEmpty()를 통해 큐가 비어 있다면 에러 메세지를 출력한다.
- '#'(샵) 을 입력했을 때
  - showQueueSize()를 실행한다.
  - showQueueSize()는 CircularArrayQueue 의 size()를 실행하여 원소의 갯수를 리턴 받은 뒤 이것을 AppView 의 outputQueueSize()를 통하여 출력한다.
- '/'(슬래시) 를 입력했을 때
  - showAll()를 실행한다.
  - showAll()는 큐의 사이즈만큼 횟수를 반복하며 CircularArrayQueue 의 elementAt()를 실행하고 이것을 AppView 의 outputElement()를 통하여 출력한다.
- '^'(곡절 부호) 를 입력했을 때
  - showFrontElement()를 실행한다.
  - showFrontElement()는 CircularArrayQueue 의 frontElement()를 실행하여 큐 내부의 맨 앞 원소를 리턴받은 뒤 AppView 의 outputFrontElement()를 통하여 출력한다.
- '!'(느낌표) 를 입력했을 때
  - conclusion()을 실행한다.
  - conclusion()은 큐의 사이즈만큼 반복하며 removeOne()을 이용하여 큐 내부의 원소를 삭제하며 삭제된 원소가 무엇인지 출력한다.
  - AppView 의 outputResult()를 이용하여 입력 받은 문자 갯수, 무시된 문자 갯수, 큐에 추가된 문자 갯수를 출력한다.
  - 프로그램을 종료한다.

2 프로그램 장단점 분석

- 장점
  - 환형(끝이 이어져있는 둥근 원형태)의 Queue 를 구현하여 얼마나 입력될지 모르는 상태에서 처음부터 많은 메모리공간을 배열형태로 선언할 필요가 없다.
  - FIFO(선입선출) 형태의 Queue 를 구현해볼 수 있다.
  - %(나머지계산)을 이용하여 배열을 환 형태의 Queue 로 구현할 수 있다.

- 단점
  - 환 형태의 배열을 구현하는 데에 약간의 이해도가 필요하다.
  - 중복 검사를 할 수 없다.
  - Queue 의 특성상 무조건 먼저 들어온 것은 먼저 나가야 한다.(First-In-First-Out)
    - 삭제를 할 때 배열의 맨 앞의 것을 삭제해야한다.
    - 만일 환 형태의 배열이 아니라면, 앞의 것을 삭제하면 뒤에 것을 하나하나씩 당겨주어야 한다.
    - 그렇지 않으면 앞에는 남아있지만 쓸 수 없는 배열 공간이 발생하므로 메모리 낭비를 하게 된다.

3 실행 결과 분석

1 입력과 출력

프로그램 실행

```
> 프로그램을 시작합니다.
[큐 입력을 시작합니다.]
```

입출력 1-1 - 영문자 입력 및 큐 삽입

```
- 문자를 입력하시오 : A
[EnQueue] 삽입된 원소는 A 입니다
```

입출력 1-2 - 영문자 입력 및 큐가 꽉 찼을 때의 출력

```
- 문자를 입력하시오 : z
ERROR : 큐가 꽉 차서 삽입이 불가능합니다.
```

입출력 2 - '#'(샵) 입력 및 큐의 내용물 총 갯수 출력

```
- 문자를 입력하시오 : #
[Size] 큐에는 현재 2개의 원소가 있습니다.
```

입출력 3-1 – '-'(하이픈) 입력 및 큐에 삭제할 원소가 없을 때

```
- 문자를 입력하시오 : -
[Empty] 큐에 원소가 없습니다.
```

입출력 3-2 - '-'(하이픈) 입력 및 큐에 삭제할 원소가 있을 때 삭제된 원소 출력

```
- 문자를 입력하시오 : -
[DeQueue] 삭제된 원소는 A 입니다
```

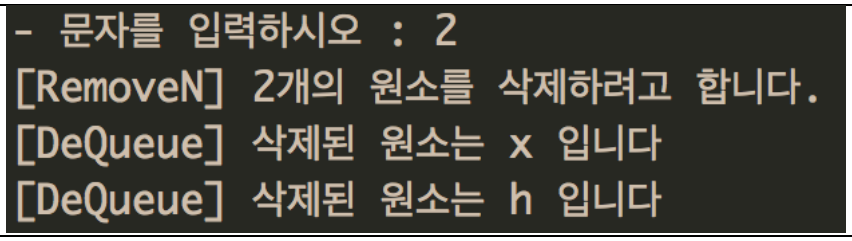
입출력 4 - '/'(슬래시) 입력 및 큐에 들어있는 원소를 차례로 출력

```
- 문자를 입력하시오 : /
[Queue] <Front> A x h <Rear>
```

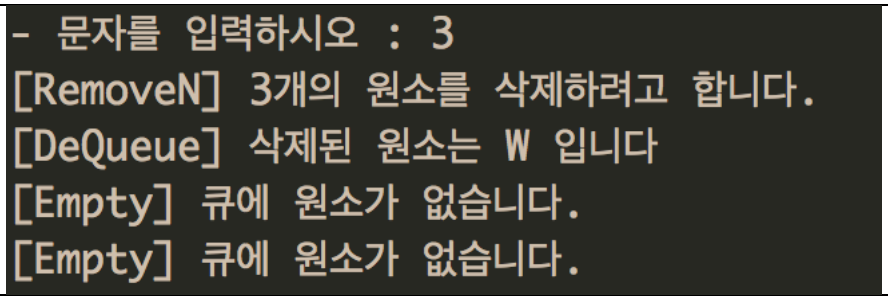
입출력 5 - '^'(곡절 부호) 입력 및 큐의 맨 앞의 원소 출력

```
- 문자를 입력하시오 : ^
[Front] 맨 앞 원소는 x 입니다
```

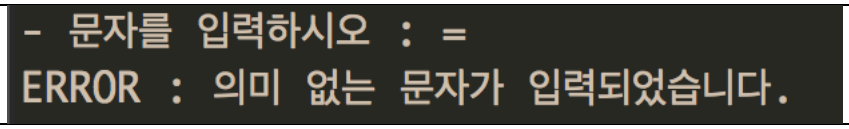
입출력 6-1 - 숫자 입력 및 숫자 개수 만큼의 원소 삭제



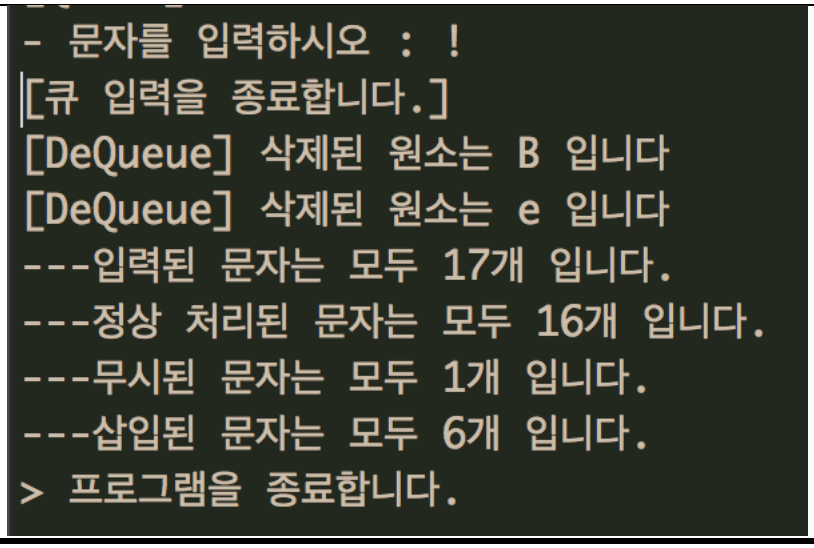
입출력 6-2 - 숫자 입력 및 숫자 개수 만큼의 원소가 존재하지 않을 때.



입출력 7 - 의미 없는 문자가 입력되었을 때



입출력 8 - '!'(느낌표) 입력 및 큐 내부 원소 삭제, 입력받은 문자, 무시된 문자, 삽입된 문자 개수 출력 및 프로그램 종료



2 결과 분석

- First-In-First-Out 개념의 Queue 자료구조를 구현하여 원소를 추가하면 차곡차곡 쌓이고, 원소를 꺼내려면 제일 앞(아래) 원소를 꺼내는 형태의 프로그램을 구현하였다.
- 한 가지의 큐를 구현해놓고, 제네릭 타입으로 선언하여 똑같은 구조의 여러개의 다른 자료형의 큐를 구현할 수 있다.
- Queue 를 환형(시작점과 끝점이 이어져 있는 둥근 원 형태)로 구현한 이유는?
  - 환형이 아닌 일반 배열을 이용하여 구현할 경우, Queue 의 특성상 먼저 들어온 것은 먼저 나가야 하므로 배열의 앞 부분을 없애 주어야 한다.
  - 이때 배열은 메모리공간이 한정적이기도 하고, 효율적인 Queue 사용을 위해서

배열 내 원소들을 한 칸 씩 앞으로 당겨 주어야 하는데, 배열의 크기가 크면 클수록 시간이 앞으로 당기는 시간이 오래 걸린다.

- 또한 당기지 않고 내버려두면 데이터는 뒤에만 있고 앞 부분에는 비어있는 채로 쓰지는 않는 상태가 발생하여 메모리 낭비를 하게 된다.
- 앞부분과 뒷부분을 이어 놓으면 원소를 삭제하고 추가를 하여도 일정부분 이상 진행 하다보면 삭제한 앞 앞부분에 원소를 추가하게 되어 메모리 낭비를 하지 않아도 된다.

● 구현방법?

- 배열을 만들어놓고, 배열의 위치값을 참조할 때 %(나머지계산)을 이용한다.
- 예를 들어 최대용량이 5 이고, 현재 값이 1 일때 1 을 5 로 나누어 준다.
- 이렇게 된다면 나머지 값은 1 이된다.
- 현재값이 2 일 때 나머지 값은 2
- 현재값이 3 일 때 나머지 값은 3
- 현재값이 4 일 때 나머지 값은 4
- 현재값이 5 일때 나머지 최대용량 5 로 나누어지므로 나머지값은 0 이며 배열의 맨 앞자리인 0 번을 참조하게 된다.
- 이렇게 한다면 0 부터 4 까지 최대크기 5 인 배열을 모두 참조하게 되므로
- 시작점과 끝점이 붙어있는, 반복할 수 있는 형태의 배열을 만들 수 있게 된다.

● 위치 계산 법

- 삽입 시
  - i 맨 뒤의 값에 + 1 을 한 값을 최대용량으로 나누어 나온 나머지 값을 다시 맨 뒤 값으로 정한다.
  - ii 이렇게 변한 rear 값을 참조값으로하여 배열에 원소를 삽입한다.
  - iii 삽입할 때 마다 +1 씩 증가한다.
  - iv _rear = (_rear + 1) % _capacity
- 삭제 시
  - i 맨 앞의 값에 +1 을 한 값을 최대 용량으로 나누어 나온 나머지 값을 다시 맨 앞의 값으로 정한다.
  - ii 이렇게 변한 front 값을 참조값으로 하여 배열의 원소를 삭제한다.
  - iii 삭제 후 해당 front 값의 위치는 null 이 되며, 이때 front 값은 배열내 제일 앞 원소의 직전 위치를 가리키게 된다.
  - iv 삭제할 때 마다 +1 씩 증가한다.
  - v _front = (_front + 1) % _capacity
- 배열이 꽉 찼는지 검사 할 때
  - i 현재 맨 뒤의 값의 +1 한 값이 _front 값과 같다면 꽉 찬 상태.
  - ii rear 값은 삽입을 하기 전에 증가하므로 맨 첫 삽입은 배열의 1 번자리부터 시작.
  - iii 계속 삽입을 하여 환형 큐를 돌아 0 번을 참조할 수 있다.
  - iv 삭제를 실행하지 않아 _front 값이 0 이라는 전제하에, rear 값이 0 이 된다면, 이것은 큐가 비어있는 것을 확인하는 isEmpty()의 내용과 같다.
  - v 배열에 데이터가 꽉 차있지만 비어있다고 인식하는 것을 방지하기 위해, 만일 rear +1 의 값이 front 와 같을때 꽉 차 있는 것으로 계산을

해야한다.

vi 이렇게 했을 때 배열의 맨 앞자리인 0 번자리는 계속해서 비어있는 상태로 되지만 결국 환형 큐를 사용할 수 있게된다.

- 결론
  - Queue 구조를 구현해볼 수 있다.
  - 환 형태의 Queue 구조를 구현해볼 수 있다.
  - 환 형태의 Queue 구조를 구현하기 위해서는 배열의 위치 참조 계산하는 법을 고려해야한다.