컴퓨터 프로그래밍 2
-hw13-

학번 : 201602038
제출일 : 2017.6.12.
이름 : 이 미 진

1. 함수 설명

1-1. 연결 리스트로 구현한 큐

1-1) AppIO

1) AppIO* AppIO_new(): AppIO 객체 생성
2) void AppIO_delete(AppIO* _this): 객체 소멸
3) char AppIO_in_nextInputChar(AppIO* _this): 문자 입력
4) char getcharDirectlyFromKeyboard(void): 키보드에서 문자를 바로 받을 수 있도록 하는 함수
5) void AppIO_out_msg_startMessage(AppIO* _this): 시작 메시지 출력
6) void AppIO_out_msg_endMessage(AppIO* _this): 종료 메시지 출력
7) void AppIO_out_msg_queueIsFull(AppIO* _this, char aChar): 큐가 꽉 찼을 때 메시지 출력
8) void AppIO_out_msg_addedElementInQueue(AppIO* _this, char aChar): 큐에 원소가 삽입됐을 때 메시지 출력
9) void AppIO_out_msg_noElementInQueue(AppIO* _this): 큐가 비었을 때 메시지 출력
10) void AppIO_out_msg_removedElementFromQueue(AppIO* _this, char aRemovedChar): 큐에서 원소가 삭제됐을 때 메시지 출력
11) void AppIO_out_FrontInQueue(AppIO* _this, Element aChar): front 원소 출력
12) void AppIO_out_msg_queueSize(AppIO* _this, int queueSize): queue size return
13) void AppIO_out_msg_Esc(AppIO* _this, Element aRemovedChar): esc 입력 시 삭제된 원소 출력
14) void AppIO_out_msg_ignoreChars(AppIO* _this): 의미없는 문자가 입력되었을 때 메시지 출력
15) void AppIO_out_labelOfFront(AppIO* _this): <Front> 출력
16) void AppIO_out_labelOfRear(AppIO* _this): <Rear> 출력
17) void AppIO_out_elementInQueue(AppIO* _this,char anElemet): 큐에 있는 원소 출력
18) void AppIO_out_msg_numberOfInputChars(AppIO* _this, int numberOfInputChars): 입력된 문자 수 출력
19) void AppIO_out_msg_numberOfNormalChars(AppIO* _this, int numberOfNormalChars): 정상적으로 처리된 문자 수 출력
20) void AppIO_out_msg_numberOfIgnoredChars(AppIO* _this, int numberOfIgnoredChars): 무시된 문자 수 출력
21) void AppIO_out_msg_numberOfAddedChars(AppIO* _this, int numberOfAddedChars): 삽입 된 문자 수 출력

1-2) AppController

1) AppController* AppController_new(void): 객체 생성
2) void AppController_delete(AppController* _this): 객체 소멸
3) void AppController_run(AppController* _this): 프로그램 실행
4) void AppController_add(AppController* _this, char aChar) : 큐에 원소 삽입
5) void AppController_removeN(AppController* _this, char aDigit): 숫자 입력시 원소 삭제
6) void AppController_remove1(AppController* _this) : - 입력시 원소 삭제
7) void AppController_showSize(AppController* _this): 큐의 크기를 보여줌

8) void AppController_showAllFromFront(AppController* _this): front부터 큐에 있는 모든 원소를 보여줌

9) void AppController_showFront(AppController* _this) : front 원소를 보여줌

10) void AppController_ignore(AppController* _this): 의미 없는 문자 무시

11) void AppController_esc(AppController* _this): Esc 입력시 종료

12) void AppController_initCharCounts(AppController* _this): 각 문자 수를 0으로 초기화 시킴

13) void AppController_countInput(AppController* _this): 입력된 문자의 개수를 증가시킴

14) void AppController_countIgnored(AppController* _this) : 무시된 문자의 개수를 증가시킴

15) void AppController_countAdded(AppController* _this) : 삽입된 문자의 개수를 증가시킴

1-3) Queue

1) Queue* Queue_new(void): Queue 객체 생성

2)void Queue_delete(Queue* _this): Queue 객체 소멸

3) Boolean Queue_isEmpty(Queue* _this): 큐가 empty이면 TRUE,아니면 FALSE

4) Boolean Queue_isFull(Queue* _this): 큐가 full 이면 TRUE를, 아니면, FALSE

5) int Queue_size(Queue* _this): 큐의 길이, 즉 큐가 가지고 있는 원소의 개수

6) Boolean Queue_add(Queue* _this, Element anElement): 큐의 rear에 item을 삽입

7) Element Queue_remove(Queue* _this): 큐의 front에서 Element를 삭제하고 그 값을 얻는다

8) void Queue_deleteLinkedChain(Queue* _this): 큐의 크기만큼 삭제

1-4) Node

1) Node* Node_new(): 노드 객체 생성

2) void Node_delete(): 노드 객체 소멸

3) void Node_setElement(Node* _this, Element newElement): 노드 element의 설정자

4) Element Node_element(Node* _this): 노드의 원소 리턴

5) void Node_setNext(Node* _this, Node* newNext): 노드 next의 설정자

6) Node* Node_next(Node* _this): 노드의 next 리턴

1-2. 배열로 구현한 큐

2-1) AppIO

1) AppIO* AppIO_new(): AppIO 객체 생성

2) void AppIO_delete(AppIO* _this): 객체 소멸

3) char AppIO_in_nextInputChar(AppIO* _this): 문자 입력

4) char getcharDirectlyFromKeyboard(void): 키보드에서 문자를 바로 받을 수 있도록 하는 함수

5) void AppIO_out_msg_startMessage(AppIO* _this): 시작 메시지 출력

6) void AppIO_out_msg_endMessage(AppIO* _this): 종료 메시지 출력

7) void AppIO_out_msg_queueIsFull(AppIO* _this, char aChar): 큐가 꽉 찼을 때 메시지 출력

8) void AppIO_out_msg_addedElementInQueue(AppIO* _this, char aChar): 큐에 원소가 삽입됐을 때 메시지 출력

9) void AppIO_out_msg_noElementInQueue(AppIO* _this): 큐가 비었을 때 메시지 출력

10) void AppIO_out_msg_removedElementFromQueue(AppIO* _this, char aRemovedChar): 큐에서 원소가 삭제됐을 때 메시지 출력

11) void AppIO_out_FrontInQueue(AppIO* _this, Element aChar): front 원소 출력

12) void AppIO_out_msg_queueSize(AppIO* _this, int queueSize): queue size return
13) void AppIO_out_msg_Esc(AppIO* _this, Element aRemovedChar): esc 입력 시 삭제된 원소 출력
14) void AppIO_out_msg_ignoreChars(AppIO* _this): 의미없는 문자가 입력되었을 때 메시지 출력
15) void AppIO_out_labelOfFront(AppIO* _this): <Front> 출력
16) void AppIO_out_labelOfRear(AppIO* _this): <Rear> 출력
17) void AppIO_out_elementInQueue(AppIO* _this,char anElemet): 큐에 있는 원소 출력
18) void AppIO_out_msg_numberOfInputChars(AppIO* _this, int numberOfInputChars): 입력된 문자 수 출력
19) void AppIO_out_msg_numberOfNormalChars(AppIO* _this, int numberOfNormalChars): 정상적으로 처리된 문자 수 출력
20) void AppIO_out_msg_numberOfIgnoredChars(AppIO* _this, int numberOfIgnoredChars): 무시된 문자 수 출력
21) void AppIO_out_msg_numberOfAddedChars(AppIO* _this, int numberOfAddedChars): 삽입 된 문자 수 출력

2-2) AppController

1) AppController* AppController_new(void): 객체 생성
2) void AppController_delete(AppController* _this): 객체 소멸
3) void AppController_run(AppController* _this): 프로그램 실행
4) void AppController_add(AppController* _this, char aChar) : 큐에 원소 삽입
5) void AppController_removeN(AppController* _this, char aDigit): 숫자 입력시 원소 삭제
6) void AppController_remove1(AppController* _this) : - 입력시 원소 삭제
7) void AppController_showSize(AppController* _this): 큐의 크기를 보여줌
8) void AppController_showAllFromFront(AppController* _this): front부터 큐에 있는 모든 원소를 보여줌
9) void AppController_showFront(AppController* _this) : front 원소를 보여줌
10) void AppController_ignore(AppController* _this): 의미 없는 문자 무시
11) void AppController_esc(AppController* _this): Esc 입력시 종료
12) void AppController_initCharCounts(AppController* _this): 각 문자 수를 0으로 초기화 시킴
13) void AppController_countInput(AppController* _this): 입력된 문자의 개수를 증가시킴
14) void AppController_countIgnored(AppController* _this) : 무시된 문자의 개수를 증가시킴
15) void AppController_countAdded(AppController* _this) : 삽입된 문자의 개수를 증가시킴

2-3) Queue

1) Queue* Queue_new(void): Queue 객체 생성Queue
2)void Queue_delete(Queue* _this): Queue 객체 소멸
3) Boolean Queue_isEmpty(Queue* _this): 큐가 empty이면 TRUE,아니면 FALSE
4) Boolean Queue_isFull(Queue* _this): 큐가 full 이면 TRUE를, 아니면, FALSE
5) int Queue_size(Queue* _this): 큐의 길이, 즉 큐가 가지고 있는 원소의 개수
6) Boolean Queue_add(Queue* _this, Element anElement): 큐의 rear에 item을 삽입
7) Element Queue_remove(Queue* _this): 큐의 front에서 Element를 삭제하고 그 값을 얻는다
8) void Queue_deleteLinkedChain(Queue* _this): 큐의 크기만큼 삭제

2. 전체 코드

2-1. 연결리스트로 구현한 큐

1-1) main.c

```
//
//  main.c
//  CP2_WEEK13
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include <stdio.h>
#include "AppController.h"

// LinkedList Queue
int main(void) {
    AppController* appController = AppController_new(); // 객체 생성
    AppController_run(appController);   // 프로그램 실행
    AppController_delete(appController);    // 객체 소멸

    return 0;
}
```

1-2) AppIO.c

```
//
//  AppIO.c
//  CP2_WEEK13
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "AppIO.h"
#include <stdio.h>
#include <unistd.h>
#include <termios.h>

struct _AppIO{

};

AppIO* AppIO_new(){ // AppIO 객체 생성
    AppIO* _this = NewObject(AppIO);
    return _this;
}

void AppIO_delete(AppIO* _this){    // 객체 소멸
```

```
    free(_this);
}

void AppIO_out_msg_startMessage(AppIO* _this){  // 시작 메시지 출력
    printf(">>> 프로그램을 시작합니다 <<<\n");
}

void AppIO_out_msg_endMessage(AppIO* _this){    //   종료 메시지 출력
    printf("<<< 프로그램을 종료합니다 >>>\n");
}

char AppIO_in_nextInputChar(AppIO* _this) { // 문자를 입력 받음
    printf(">>> 문자를 입력하세요 : ");
    char inputChar = getcharDirectlyFromKeyboard();
    printf("\n");
    return inputChar;
}

char getcharDirectlyFromKeyboard(void) {
    struct termios oldAttr;
    struct termios newAttr;
    char charFromKeyboard;

    fpurge(stdin);  // stdin buffer를 비운다
    tcgetattr( STDIN_FILENO, &oldAttr);
    newAttr = oldAttr;
    newAttr.c_lflag &= ~(ICANON | ECHO );
    tcsetattr(STDIN_FILENO, TCSANOW, &newAttr);
    charFromKeyboard = getchar();
    tcsetattr(STDIN_FILENO, TCSANOW, &newAttr);

    return charFromKeyboard;

}
void AppIO_out_msg_queueIsFull(AppIO* _this, char aChar) {   // 큐
가 꽉 찼을 때 메시지 출력
    printf("\n[Full] 큐가 꽉 차서 원소 \'%c\' 는 삽입이 불가능합니다.
\n",aChar);
}

void AppIO_out_msg_addedElementInQueue(AppIO* _this, char aChar) {
// 큐에 원소가 삽입됐을 때 메시지 출력
    printf("\n[Add] 삽입된 원소는 \'%c\' 입니다.\n",aChar);
}

void AppIO_out_msg_noElementInQueue(AppIO* _this) { // 큐가 비었을 때
메시지 출력
    printf("\n[Empty] 큐에 삭제할 원소가 없습니다.\n");
}
```

```c
void AppIO_out_msg_removedElementFromQueue(AppIO* _this, char
aRemovedChar) {    // 큐에서 원소가 삭제됐을 때 메시지 출력
    printf("\n[Remove1] 삭제된 원소는 \'%c\' 입니다. \n",
aRemovedChar);
}

void AppIO_out_msg_queueSize(AppIO* _this, int queueSize) {
    printf("\n[Size] 현재 큐의 크기는 %d 입니다.\n",queueSize);
}

void AppIO_out_msg_Esc(AppIO* _this, Element aRemovedChar){ // esc
입력 시 삭제된 원소 출력
    printf("\n[Esc] 삭제된 원소는 \'%c\' 입니다. \n", aRemovedChar);
}
void AppIO_out_msg_ignoreChars(AppIO* _this){   // 의미없는 문자가 입력
되었을 때 메시지 출력
    printf("\n[Ignore] 의미 없는 문자가 입력되었습니다.\n");
}
void AppIO_out_FrontInQueue(AppIO* _this, Element aChar){   //
front 원소 출력
    printf("\n[Front] Front 원소는 \'%c\' 입니다. \n", aChar);
}
void AppIO_out_labelOfFront(AppIO* _this){  // <Front> 출력
    printf("<Front>");
}

void AppIO_out_labelOfRear(AppIO* _this) {   // <Rear> 출력
    printf("<Rear>\n");
}

void AppIO_out_elementInQueue(AppIO* _this,char anElemet) {
    printf("%c ",anElemet);
}

void AppIO_out_msg_numberOfInputChars(AppIO* _this, int
numberOfInputChars){    //  입력된 문자 수 출력
    printf(">>> 입력된 문자는 %d 개 입니다.\n",numberOfInputChars);
}
void AppIO_out_msg_numberOfNormalChars(AppIO* _this, int
numberOfNormalChars){ // 정상적으로 처리된 문자 수 출력
    printf(">>> 정상적으로 처리된 문자는 %d 개 입니다.
\n",numberOfNormalChars);
}
void AppIO_out_msg_numberOfIgnoredChars(AppIO* _this, int
numberOfIgnoredChars){    // 무시된 문자 수 출력
    printf(">>> 무시된 문자는 %d 개 입니다.\n",numberOfIgnoredChars);
}
void AppIO_out_msg_numberOfAddedChars(AppIO* _this, int
numberOfAddedChars){   // 삽입 된 문자 수 출력
    printf(">>> 삽입된 문자는 %d 개 입니다.\n",numberOfAddedChars);
```

```
}
```

1-3) AppIO.h

```
//
//  AppIO.h
//  CP2_WEEK13
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppIO_h
#define AppIO_h
#include "Common.h"
#include "Queue.h"

typedef struct _AppIO AppIO;

AppIO* AppIO_new(); // AppIO 객체 생성
void AppIO_delete(AppIO* _this);    // 객체 소멸
char AppIO_in_nextInputChar(AppIO* _this);   // 문자 입력
char getcharDirectlyFromKeyboard(void); // 키보드에서 문자를 바로 받을 수 있도록 하는 함수
void AppIO_out_msg_startMessage(AppIO* _this);  // 시작 메시지 출력
void AppIO_out_msg_endMessage(AppIO* _this);    //   종료 메시지 출력
void AppIO_out_msg_queueIsFull(AppIO* _this, char aChar);   // 큐가 꽉 찼을 때 메시지 출력
void AppIO_out_msg_addedElementInQueue(AppIO* _this, char aChar);
// 큐에 원소가 삽입됐을 때 메시지 출력
void AppIO_out_msg_noElementInQueue(AppIO* _this);  // 큐가 비었을 때 메시지 출력
void AppIO_out_msg_removedElementFromQueue(AppIO* _this, char aRemovedChar);    // 큐에서 원소가 삭제됐을 때 메시지 출력
void AppIO_out_FrontInQueue(AppIO* _this, Element aChar);   // front 원소 출력
void AppIO_out_msg_queueSize(AppIO* _this, int queueSize);  // queue size return
void AppIO_out_msg_Esc(AppIO* _this, Element aRemovedChar); // esc 입력 시 삭제된 원소 출력
void AppIO_out_msg_ignoreChars(AppIO* _this);   // 의미없는 문자가 입력되었을 때 메시지 출력
void AppIO_out_labelOfFront(AppIO* _this);  // <Front> 출력
void AppIO_out_labelOfRear(AppIO* _this);   // <Rear> 출력
void AppIO_out_elementInQueue(AppIO* _this,char anElemet);  // 큐에 있는 원소 출력

// 처리 후 종료
```

```
void AppIO_out_msg_numberOfInputChars(AppIO* _this, int
numberOfInputChars);    //  입력된 문자 수 출력
void AppIO_out_msg_numberOfNormalChars(AppIO* _this, int
numberOfNormalChars);  // 정상적으로 처리된 문자 수 출력
void AppIO_out_msg_numberOfIgnoredChars(AppIO* _this, int
numberOfIgnoredChars);    // 무시된 문자 수 출력
void AppIO_out_msg_numberOfAddedChars(AppIO* _this, int
numberOfAddedChars);    // 삽입 된 문자 수 출력

#endif /* AppIO_h */
```

1-4) AppController.c

```
//
//  AppController.c
//  CP2_WEEK13
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "AppController.h"
#define Esc 27

// 문자 별 행위를 처리
void AppController_add(AppController* _this, char aChar);   // 큐에
원소 삽입
void AppController_removeN(AppController* _this, char aDigit);  //
숫자 입력시 원소 삭제
void AppController_remove1(AppController* _this);   // - 입력시 원소
삭제
void AppController_showSize(AppController* _this); // 큐의 크기를 보여
줌
void AppController_showAllFromFront(AppController* _this);  //
front부터 큐에 있는 모든 원소를 보여줌
void AppController_showFront(AppController* _this); // front 원소를
보여줌
void AppController_ignore(AppController* _this);   // 의미 없는 문자
무시
void AppController_esc(AppController* _this);  // Esc 입력시 종료

// 행위 별 문자 수를 센다
void AppController_initCharCounts(AppController* _this);    //각 문
자 수를 0으로 초기화 시킴
void AppController_countInput(AppController* _this);    // 입력된 문
자의 개수를 증가시킴
void AppController_countIgnored(AppController* _this);  // 무시된 문
자의 개수를 증가시킴
```

```
void AppController_countAdded(AppController* _this);    // 삽입된 문
자의 개수를 증가시킴

struct _AppController {
    AppIO* _appIO;
    Queue* _queue;
    int _inputChars;    // 입력된 문자의 개수
    int _ignoredChars;  // 무시된 문자의 개수
    int _addedChars;    // 삽입된 문자의 개수
};

AppController* AppController_new(void) { // 객체 생성
    AppController* _this;
    _this = NewObject(AppController);

    _this->_appIO = AppIO_new();
    _this->_queue = Queue_new();
    _this->_inputChars = 0;
    _this->_ignoredChars = 0;
    _this->_addedChars = 0;

    return _this;
}

void AppController_delete(AppController* _this){    // 객체 소멸
    Queue_delete(_this->_queue);
    AppIO_delete(_this->_appIO);
    free(_this);
}

void AppController_add(AppController* _this, char aChar) {  // 큐에
원소 삽입
    if( Queue_isFull(_this->_queue)) {
        AppIO_out_msg_queueIsFull(_this->_appIO, aChar);
    }
    else {
        Queue_add(_this->_queue, aChar);
        AppController_countAdded(_this);
        AppIO_out_msg_addedElementInQueue(_this->_appIO,aChar);
    }
}

void AppController_removeN(AppController* _this, char aDigit){
    int numberOfCharsRemoved = aDigit-'0';  // digit 문자를 숫자로
    if( Queue_isEmpty(_this->_queue)) {
        AppIO_out_msg_noElementInQueue(_this->_appIO);
    }
    else {
        numberOfCharsRemoved = Queue_remove(_this->_queue);
```

```
        AppIO_out_msg_removedElementFromQueue(_this->_appIO,
numberOfCharsRemoved);
    }
}

void AppController_remove1(AppController* _this) {  // 큐에서 원소 삭
제
    char removedChar;
    if( Queue_isEmpty(_this->_queue)) {
        AppIO_out_msg_noElementInQueue(_this->_appIO);
    }
    else {
        removedChar = Queue_remove(_this->_queue);
        AppIO_out_msg_removedElementFromQueue(_this-
>_appIO,removedChar);
    }
}

void AppController_showAllFromFront(AppController* _this) {
    AppIO_out_labelOfFront(_this->_appIO);
    for( int i=0; i< Queue_size(_this->_queue); i++ ) {
        AppIO_out_elementInQueue(_this->_appIO,
Queue_elementAt(_this->_queue, i));
    }
    AppIO_out_labelOfRear(_this->_appIO);
}

void AppController_showSize(AppController* _this){ // 큐의 크기를 보여
줌
    AppIO_out_msg_queueSize(_this->_appIO, Queue_size(_this-
>_queue));
}

void AppController_showFront(AppController* _this){ // front 원소를
보여줌
    AppIO_out_FrontInQueue(_this->_appIO, Queue_elementAt(_this-
>_queue, 0));
}

void AppController_ignore(AppController* _this) {
    AppIO_out_msg_ignoreChars(_this->_appIO);
    AppController_countIgnored(_this);
}
void AppController_initCharCounts(AppController* _this) {   //각 문
자 수를 0으로 초기화 시킴
    _this->_inputChars = 0; // 입력된 문자의 개수
    _this->_ignoredChars = 0;  // 무시된 문자의 개수
    _this->_addedChars = 0;   // 삽입된 문자의 개수
}
```

```
void AppController_countInput (AppController* _this) {
    _this->_inputChars++; // 입력된 문자의 개수 증가
}

void AppController_countIgnored(AppController* _this){
    _this->_ignoredChars++;  // 무시된 문자의 개수 증가
}

void AppController_countAdded(AppController* _this){
    _this->_addedChars++;   // 삽입된 문자의 개수 증가
}

void AppController_esc(AppController* _this) {  // Esc 입력시 종료
    AppController_removeN(_this, Queue_size(_this->_queue)+'0');
    AppIO_out_msg_numberOfInputChars(_this->_appIO, _this-
>_inputChars);
    AppIO_out_msg_numberOfNormalChars(_this->_appIO, (_this-
>_inputChars)-(_this->_ignoredChars));
    AppIO_out_msg_numberOfIgnoredChars(_this->_appIO, _this-
>_ignoredChars);
    AppIO_out_msg_numberOfAddedChars(_this->_appIO, _this-
>_addedChars);
}

// 전체 제어
void AppController_run(AppController* _this){
    AppController_initCharCounts(_this);

    AppIO_out_msg_startMessage(_this->_appIO);

 //   AppIO_out_msg_inputChar(_this->_appIO);
    char inputChar = AppIO_in_nextInputChar(_this->_appIO);
    AppController_countInput(_this);

    while( inputChar != Esc ) {
        if( isAlpha(inputChar) ) {
            AppController_add(_this, inputChar);
        }
        else if( isDigit(inputChar)) {
            AppController_removeN(_this, inputChar);
        }
        else if( inputChar == '-' ) {
            AppController_remove1(_this);
        }
        else if( inputChar == '#' ) {
            AppController_showSize(_this);
        }
        else if( inputChar == '/' ) {
            AppController_showAllFromFront(_this);
        }
        else if( inputChar == '^' ) {
```

```
            AppController_showFront(_this);
        }
        else {
            AppController_ignore(_this);
        }

        inputChar = AppIO_in_nextInputChar(_this->_appIO);
        AppController_countInput(_this);
    }
    AppController_esc(_this);

    AppIO_out_msg_endMessage(_this->_appIO);
}
```

1-5) AppController.h

```
//
//  AppController.h
//  CP2_WEEK13
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppController_h
#define AppController_h

#include "AppIO.h"
#include "Common.h"
#include "Queue.h"

typedef struct _AppController AppController;

AppController* AppController_new(void); // 객체 생성
void AppController_delete(AppController* _this);    // 객체 소멸
// 전체 제어
void AppController_run(AppController* _this);   // 프로그램 실행

#endif /* AppController_h */
```

1-6) Queue.c

```
//
//  Queue.c
//  CP2_WEEK13
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//
```

```c
#include "Queue.h"

struct _Queue {
    Node* _front;
    Node* _rear;
    int _size;
};

// Queue 객체의 생성과 소멸
Queue* Queue_new(void){ // Queue 객체의 생성
    Queue* _this;
    _this = NewObject(Queue);
    _this->_front = NULL;
    _this->_rear = NULL;
    _this->_size = 0;

    return _this;
}

void Queue_delete(Queue* _this) {   // queue  객체 소멸
    Queue_deleteLinkedChain(_this);
    free(_this);
}

// Queue 객체의 점검
Boolean Queue_isEmpty(Queue* _this){    // 큐가 empty이면 TRUE,아니면
FALSE
    return _this->_front == NULL;
}

Boolean Queue_isFull(Queue* _this){ // 큐가 full 이면 TRUE를, 아니면,
FALSE
    return FALSE;
}

int Queue_size(Queue* _this){   // 큐의 길이, 즉 큐가 가지고 있는 원소의 개
수
    return _this->_size;
}

// Queue 객체에 지시
Boolean Queue_add(Queue* _this, Element anElement){ // 큐의 rear에
item을 삽입
    Node* addedNode = Node_new();
    Node_setElement(addedNode, anElement);

    if( !Queue_isEmpty(_this)) {
        Node_setNext(_this->_rear, addedNode);
        _this->_rear = addedNode;
```

```
        _this->_size++;

        return TRUE;
    }
    else {
        _this->_front = addedNode;
        _this->_rear = addedNode;
        _this->_size++;

        return TRUE;
    }
}

Element Queue_remove(Queue* _this){ // 큐의 front에서 Element를 삭제하
고 그 값을 얻는다
    if( _this->_size > 1 ) {
        Node* removedNode = _this->_front;
        Element removedElement = Node_element(_this->_front);
        _this->_front = Node_next(_this->_front);

        Node_delete(removedNode);
        _this->_size--;

        return removedElement;
    }
    else {
        Element removedElement = Node_element(_this->_front);
        Node_delete(_this->_front);

        _this->_front = NULL;
        _this->_rear = NULL;
        _this->_size--;

        return removedElement;
    }
}

void Queue_deleteLinkedChain(Queue* _this){ // 큐의 크기만큼 삭제
    int i;
    for( i=1; i < _this->_size; i++ ) {
        Queue_remove(_this);
    }
}

Element Queue_elementAt(Queue* _this, int aPosition){   //큐의
aPosition 번째 원소를 얻는다.
    Node* currentNode = _this->_front;
    int i;
    for( i=0; i < aPosition; i++ ) {
        currentNode = Node_next(currentNode);
    }
```

```
    return Node_element(currentNode);
}
```

1-7) Queue.h

```
//
//  Queue.h
//  CP2_WEEK13
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Queue_h
#define Queue_h

#include "Common.h"
#include "Node.h"

typedef struct _Queue Queue;

// Queue 객체의 생성과 소멸
Queue* Queue_new(void);// Queue 객체의 생성
void Queue_delete(Queue* _this);  // Queue  객체 소멸

// Queue 객체의 점검
Boolean Queue_isEmpty(Queue* _this);    // 큐가 empty이면 TRUE,아니면
FALSE
Boolean Queue_isFull(Queue* _this); // 큐가 full 이면 TRUE를, 아니면,
FALSE
int Queue_size(Queue* _this);   // 큐의 길이, 즉 큐가 가지고 있는 원소의 개
수

// Queue 객체에 지시
Boolean Queue_add(Queue* _this, Element anElement); // 큐의 rear에
item을 삽입
Element Queue_remove(Queue* _this); // 큐의 front에서 Element를 삭제하
고 그 값을 얻는다
void Queue_deleteLinkedChain(Queue* _this); // 큐의 크기만큼 삭제
Element Queue_elementAt(Queue* _this, int aPosition);   //큐의
aPosition 번째 원소를 얻는다.

#endif /* Queue_h */
```

1-8) Node.c

```
//
//  Node.c
```

```
//  CP2_WEEK13
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Node.h"

struct _Node {
    Element _element;
    Node* _next;
};

Node* Node_new() {  // 노드 객체 생성
    Node* _this = NewObject(Node);
    _this->_element = 0;
    _this->_next = NULL;

    return _this;
}

void Node_delete(Node* _this) { // 노드 객체 소멸
    free(_this);
}

void Node_setElement(Node* _this, Element newElement) {  // 노드
element의 설정자
    _this->_element = newElement;
}

Element Node_element(Node* _this){  // 노드의 원소 리턴
    return _this->_element;
}

void Node_setNext(Node* _this, Node* newNext) {  // 노드 next의 설정자
    _this->_next = newNext;
}

Node* Node_next(Node* _this) {  // 노드의 next 리턴
    return _this->_next;
}
```

1-9) Node.h

```
//
//  Node.h
//  CP2_WEEK13
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
```

```
//

#ifndef Node_h
#define Node_h

#include <stdio.h>
#include "Common.h"

typedef struct _Node Node;

Node* Node_new();   // 노드 객체 생성
void Node_delete(Node* _this); // 노드 객체 소멸
void Node_setElement(Node* _this, Element newElement);  // 노드
element의 설정자
Element Node_element(Node* _this);  // 노드의 원소 리턴

void Node_setNext(Node* _this, Node* newNext);  // 노드 next의 설정자
Node* Node_next(Node* _this);   // 노드의  next 리턴


#endif /* Node_h */
```

1-10) Common.h

```
//
//  Common.h
//  CP2_WEEK13
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Common_h
#define Common_h

#include <math.h>
#include <stdlib.h>

#define NewVector(TYPE,SIZE) (TYPE*)malloc(sizeof(TYPE)*SIZE)
#define NewObject(TYPE) (TYPE*)malloc(sizeof(TYPE))
#define isDigit(CHAR) (('0' <= CHAR) && (CHAR <= '9'))
#define isAlpha(CHAR) ( (('a' <= CHAR) && (CHAR <= 'z')) || (('A'
<= CHAR) && (CHAR <= 'Z')))
typedef enum {FALSE,TRUE} Boolean;  // FALSE와 TRUE 값을 갖는 Boolean
선언
typedef int Element;

#endif /* Common_h */
```

2-2. 배열로 구현한 큐

2-1) main.c

```
//
//  main.c
//  CP2_WEEK13_2
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

// Array Queue
#include <stdio.h>
#include "AppController.h"

int main(void) {
    AppController* appController = AppController_new(); // 객체 생성
    AppController_run(appController);   // 프로그램 실행
    AppController_delete(appController);    // 객체 소멸

    return 0;
}
```

2-2) AppIO.c

```
//
//  AppIO.c
//  CP2_WEEK13_2
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "AppIO.h"
#include <stdio.h>
#include <unistd.h>
#include <termios.h>

struct _AppIO{

};

AppIO* AppIO_new(){ // AppIO 객체 생성
    AppIO* _this = NewObject(AppIO);
    return _this;
}

void AppIO_delete(AppIO* _this){    // 객체 소멸
```

```
    free(_this);
}

void AppIO_out_msg_startMessage(AppIO* _this){  // 시작 메시지 출력
    printf(">>> 프로그램을 시작합니다 <<<\n");
}

void AppIO_out_msg_endMessage(AppIO* _this){    //   종료 메시지 출력
    printf("<<< 프로그램을 종료합니다 >>>\n");
}

char AppIO_in_nextInputChar(AppIO* _this) { // 문자를 입력 받음
    printf(">>> 문자를 입력하세요 : ");
    char inputChar = getcharDirectlyFromKeyboard();
    printf("\n");
    return inputChar;
}

char getcharDirectlyFromKeyboard(void) {
    struct termios oldAttr;
    struct termios newAttr;
    char charFromKeyboard;

    fpurge(stdin);  // stdin buffer를 비운다
    tcgetattr( STDIN_FILENO, &oldAttr);
    newAttr = oldAttr;
    newAttr.c_lflag &= ~(ICANON | ECHO );
    tcsetattr(STDIN_FILENO, TCSANOW, &newAttr);
    charFromKeyboard = getchar();
    tcsetattr(STDIN_FILENO, TCSANOW, &newAttr);

    return charFromKeyboard;

}
void AppIO_out_msg_queueIsFull(AppIO* _this, char aChar) {   // 큐
가 꽉 찼을 때 메시지 출력
    printf("\n[Full] 큐가 꽉 차서 원소 \'%c\' 는 삽입이 불가능합니다.
\n",aChar);
}

void AppIO_out_msg_addedElementInQueue(AppIO* _this, char aChar) {
// 큐에 원소가 삽입됐을 때 메시지 출력
    printf("\n[Add] 삽입된 원소는 \'%c\' 입니다.\n",aChar);
}

void AppIO_out_msg_noElementInQueue(AppIO* _this) { // 큐가 비었을 때
메시지 출력
    printf("\n[Empty] 큐에 삭제할 원소가 없습니다.\n");
}
```

```c
void AppIO_out_msg_removedElementFromQueue(AppIO* _this, char
aRemovedChar) {    // 큐에서 원소가 삭제됐을 때 메시지 출력
    printf("\n[Remove1] 삭제된 원소는 \'%c\' 입니다. \n",
aRemovedChar);
}

void AppIO_out_msg_queueSize(AppIO* _this, int queueSize) {
    printf("\n[Size] 현재 큐의 크기는 %d 입니다.\n",queueSize);
}

void AppIO_out_msg_Esc(AppIO* _this, Element aRemovedChar){ // esc
입력 시 삭제된 원소 출력
    printf("\n[Esc] 삭제된 원소는 \'%c\' 입니다. \n", aRemovedChar);
}
void AppIO_out_msg_ignoreChars(AppIO* _this){   // 의미없는 문자가 입력
되었을 때 메시지 출력
    printf("\n[Ignore] 의미 없는 문자가 입력되었습니다.\n");
}
void AppIO_out_FrontInQueue(AppIO* _this, Element aChar){   //
front 원소 출력
    printf("\n[Front] Front 원소는 \'%c\' 입니다. \n", aChar);
}
void AppIO_out_labelOfFront(AppIO* _this){  // <Front> 출력
    printf("<Front>");
}

void AppIO_out_labelOfRear(AppIO* _this) {   // <Rear> 출력
    printf("<Rear>\n");
}

void AppIO_out_elementInQueue(AppIO* _this,char anElemet) {
    printf("%c ",anElemet);
}

void AppIO_out_msg_numberOfInputChars(AppIO* _this, int
numberOfInputChars){    //  입력된 문자 수 출력
    printf(">>> 입력된 문자는 %d 개 입니다.\n",numberOfInputChars);
}
void AppIO_out_msg_numberOfNormalChars(AppIO* _this, int
numberOfNormalChars){ // 정상적으로 처리된 문자 수 출력
    printf(">>> 정상적으로 처리된 문자는 %d 개 입니다.
\n",numberOfNormalChars);
}
void AppIO_out_msg_numberOfIgnoredChars(AppIO* _this, int
numberOfIgnoredChars){    // 무시된 문자 수 출력
    printf(">>> 무시된 문자는 %d 개 입니다.\n",numberOfIgnoredChars);
}
void AppIO_out_msg_numberOfAddedChars(AppIO* _this, int
numberOfAddedChars){   // 삽입 된 문자 수 출력
    printf(">>> 삽입된 문자는 %d 개 입니다.\n",numberOfAddedChars);
```

```
}
```

2-3) AppIO.h

```
//
//  AppIO.h
//  CP2_WEEK13_2
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppIO_h
#define AppIO_h

#include "Common.h"
#include "Queue.h"

typedef struct _AppIO AppIO;

AppIO* AppIO_new(); // AppIO 객체 생성
void AppIO_delete(AppIO* _this);    // 객체 소멸
char AppIO_in_nextInputChar(AppIO* _this);   // 문자 입력
char getcharDirectlyFromKeyboard(void);
void AppIO_out_msg_startMessage(AppIO* _this);  // 시작 메시지 출력
void AppIO_out_msg_endMessage(AppIO* _this);    //   종료 메시지 출력
void AppIO_out_msg_queueIsFull(AppIO* _this, char aChar);   // 큐가
꽉 찼을 때 메시지 출력
void AppIO_out_msg_addedElementInQueue(AppIO* _this, char aChar);
// 큐에 원소가 삽입됐을 때 메시지 출력
void AppIO_out_msg_noElementInQueue(AppIO* _this);  // 큐가 비었을 때
메시지 출력
void AppIO_out_msg_removedElementFromQueue(AppIO* _this, char
aRemovedChar);    // 큐에서 원소가 삭제됐을 때 메시지 출력
void AppIO_out_FrontInQueue(AppIO* _this, Element aChar);   //
front 원소 출력
void AppIO_out_msg_queueSize(AppIO* _this, int queueSize);  //
queue size return
void AppIO_out_msg_Esc(AppIO* _this, Element aRemovedChar); // esc
입력 시 삭제된 원소 출력
void AppIO_out_msg_ignoreChars(AppIO* _this);   // 의미없는 문자가 입력
되었을 때 메시지 출력
void AppIO_out_labelOfFront(AppIO* _this);  // <Front> 출력
void AppIO_out_labelOfRear(AppIO* _this);   // <Rear> 출력
void AppIO_out_elementInQueue(AppIO* _this,char anElemet);  // 큐에
있는 원소 출력

// 처리 후 종료
```

```
void AppIO_out_msg_numberOfInputChars(AppIO* _this, int
numberOfInputChars);    //  입력된 문자 수 출력
void AppIO_out_msg_numberOfNormalChars(AppIO* _this, int
numberOfNormalChars);  // 정상적으로 처리된 문자 수 출력
void AppIO_out_msg_numberOfIgnoredChars(AppIO* _this, int
numberOfIgnoredChars);    // 무시된 문자 수 출력
void AppIO_out_msg_numberOfAddedChars(AppIO* _this, int
numberOfAddedChars);    // 삽입 된 문자 수 출력

#endif /* AppIO_h */
```

2-4) AppController.c

```
//
//  AppController.c
//  CP2_WEEK13_2
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "AppController.h"
#define Esc 27

// 문자 별 행위를 처리
void AppController_add(AppController* _this, char aChar);   // 큐에
원소 삽입
void AppController_removeN(AppController* _this, char aDigit);  //
숫자 입력시 원소 삭제
void AppController_remove1(AppController* _this);   // - 입력시 원소
삭제
void AppController_showSize(AppController* _this); // 큐의 크기를 보여
줌
void AppController_showAllFromFront(AppController* _this);  //
front부터 큐에 있는 모든 원소를 보여줌
void AppController_showFront(AppController* _this); // front 원소를
보여줌
void AppController_ignore(AppController* _this);     // 의미 없는 문자
무시
void AppController_esc(AppController* _this);  // Esc 입력시 종료

// 행위 별 문자 수를 센다
void AppController_initCharCounts(AppController* _this);    //각 문
자 수를 0으로 초기화 시킴
void AppController_countInput(AppController* _this);    // 입력된 문
자의 개수를 증가시킴
void AppController_countIgnored(AppController* _this);  // 무시된 문
자의 개수를 증가시킴
```

```
void AppController_countAdded(AppController* _this);    // 삽입된 문
자의 개수를 증가시킴

struct _AppController {
    AppIO* _appIO;
    Queue* _queue;
    int _inputChars;    // 입력된 문자의 개수
    int _ignoredChars;  // 무시된 문자의 개수
    int _addedChars;    // 삽입된 문자의 개수
};

AppController* AppController_new(void) { // 객체 생성
    AppController* _this;
    _this = NewObject(AppController);

    _this->_appIO = AppIO_new();
    _this->_queue = Queue_new(DEFAULT_QUEUE_CAPACITY);
    _this->_inputChars = 0;
    _this->_ignoredChars = 0;
    _this->_addedChars = 0;

    return _this;
}

void AppController_delete(AppController* _this){    // 객체 소멸
    Queue_delete(_this->_queue);
    AppIO_delete(_this->_appIO);
    free(_this);
}

void AppController_add(AppController* _this, char aChar) {  // 큐에
원소 삽입
    if( Queue_isFull(_this->_queue)) {
        AppIO_out_msg_queueIsFull(_this->_appIO, aChar);
    }
    else {
        Queue_add(_this->_queue, aChar);
        AppController_countAdded(_this);
        AppIO_out_msg_addedElementInQueue(_this->_appIO,aChar);
    }
}

void AppController_removeN(AppController* _this, char aDigit){
    int numberOfCharsRemoved = aDigit-'0';  // digit 문자를 숫자로
    if( Queue_isEmpty(_this->_queue)) {
        AppIO_out_msg_noElementInQueue(_this->_appIO);
    }
    else {
        numberOfCharsRemoved = Queue_remove(_this->_queue);
```

```
        AppIO_out_msg_removedElementFromQueue(_this->_appIO,
numberOfCharsRemoved);
    }
}

void AppController_remove1(AppController* _this) {  // 큐에서 원소 삭
제
    char removedChar;
    if( Queue_isEmpty(_this->_queue)) {
        AppIO_out_msg_noElementInQueue(_this->_appIO);
    }
    else {
        removedChar = Queue_remove(_this->_queue);
        AppIO_out_msg_removedElementFromQueue(_this-
>_appIO,removedChar);
    }
}

void AppController_showAllFromFront(AppController* _this) { //
Front 부터 전체 원소를 보여줌
    AppIO_out_labelOfFront(_this->_appIO);
    for( int i=0; i< Queue_size(_this->_queue); i++ ) {
        AppIO_out_elementInQueue(_this->_appIO,
Queue_elementAt(_this->_queue, i));
    }
    AppIO_out_labelOfRear(_this->_appIO);
}

void AppController_showSize(AppController* _this){ // 큐의 크기를 보여
줌
    AppIO_out_msg_queueSize(_this->_appIO, Queue_size(_this-
>_queue));
}

void AppController_showFront(AppController* _this){ // front 원소를
보여줌
    AppIO_out_FrontInQueue(_this->_appIO, Queue_elementAt(_this-
>_queue, 0));
}

void AppController_ignore(AppController* _this) {   // 의미 없는 문자
무시
    AppIO_out_msg_ignoreChars(_this->_appIO);
    AppController_countIgnored(_this);
}
void AppController_initCharCounts(AppController* _this) {   //각 문
자 수를 0으로 초기화 시킴
    _this->_inputChars = 0; // 입력된 문자의 개수
    _this->_ignoredChars = 0;  // 무시된 문자의 개수
    _this->_addedChars = 0;   // 삽입된 문자의 개수
```

```
}

void AppController_countInput (AppController* _this) {
    _this->_inputChars++; // 입력된 문자의 개수 증가
}

void AppController_countIgnored(AppController* _this){
    _this->_ignoredChars++;  // 무시된 문자의 개수 증가
}

void AppController_countAdded(AppController* _this){
    _this->_addedChars++;   // 삽입된 문자의 개수 증가
}

void AppController_esc(AppController* _this) {  // Esc 입력시 종료
    AppController_removeN(_this, Queue_size(_this->_queue)+'0');
    AppIO_out_msg_numberOfInputChars(_this->_appIO, _this-
>_inputChars);
    AppIO_out_msg_numberOfNormalChars(_this->_appIO, (_this-
>_inputChars)-(_this->_ignoredChars));
    AppIO_out_msg_numberOfIgnoredChars(_this->_appIO, _this-
>_ignoredChars);
    AppIO_out_msg_numberOfAddedChars(_this->_appIO, _this-
>_addedChars);
}


// 전체 제어
void AppController_run(AppController* _this){
    AppController_initCharCounts(_this);

    AppIO_out_msg_startMessage(_this->_appIO);

    //   AppIO_out_msg_inputChar(_this->_appIO);
    char inputChar = AppIO_in_nextInputChar(_this->_appIO);
    AppController_countInput(_this);

    while( inputChar != Esc ) {
        if( isAlpha(inputChar) ) {
            AppController_add(_this, inputChar);
        }
        else if( isDigit(inputChar)) {
            AppController_removeN(_this, inputChar);
        }
        else if( inputChar == '-' ) {
            AppController_remove1(_this);
        }
        else if( inputChar == '#' ) {
            AppController_showSize(_this);
        }
        else if( inputChar == '/' ) {
```

```
            AppController_showAllFromFront(_this);
        }
        else if( inputChar == '^' ) {
            AppController_showFront(_this);
        }
        else {
            AppController_ignore(_this);
        }

        inputChar = AppIO_in_nextInputChar(_this->_appIO);
        AppController_countInput(_this);
    }
    AppController_esc(_this);

    AppIO_out_msg_endMessage(_this->_appIO);
}
```

2-5) AppController.h

```
//
//  AppController.h
//  CP2_WEEK13_2
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppController_h
#define AppController_h

#include "AppIO.h"
#include "Common.h"
#include "Queue.h"

typedef struct _AppController AppController;

AppController* AppController_new(void); // 객체 생성
void AppController_delete(AppController* _this);    // 객체 소멸
// 전체 제어
void AppController_run(AppController* _this);

#endif /* AppController_h */
```

2-6) Queue.c

```
//
//  Queue.c
//  CP2_WEEK13_2
//
```

```
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Queue.h"
#include "Common.h"

void Queue_deleteLinkedChain(Queue* _this);

struct _Queue {
    int _capacity;  // 배열로 리스트를 구현할 때 반드시 필요한 속성
    int _front;
    int _rear;
    Element* _elements;
};

// Queue 객체의 생성과 소멸
Queue* Queue_new(int givenCapacity){    // Queue 객체의 생성
    Queue* _this;
    _this = NewObject(Queue);
    _this->_capacity = givenCapacity;
    _this->_elements = NewVector(Element, givenCapacity);
    _this->_front = 0;
    _this->_rear = 0;

    return _this;
}

void Queue_delete(Queue* _this) {   // Queue 객체의 소멸
    free(_this->_elements);
    free(_this);
}

// Queue 객체의 점검
Boolean Queue_isEmpty(Queue* _this){    // 큐가 empty이면 TRUE,아니면
FALSE
    return (_this->_front == _this->_rear);
}

Boolean Queue_isFull(Queue* _this){ // 큐가 full 이면 TRUE를, 아니면,
FALSE
    int nextRear = (_this->_rear+1) % _this->_capacity;
    return ( nextRear == _this->_front );
}

int Queue_size(Queue* _this){   // 큐의 길이, 즉 큐가 가지고 있는 원소의 개
수
    int size = 0;
    if( _this->_front < _this->_rear ){
        size = _this->_rear - _this->_front;
```

```
    }
    else if( _this->_front > _this->_rear ) {
        size = _this->_capacity - (( _this->_front - _this-
>_rear)-1);
    }
    else {
        size = 0;
    }
    return size;
}

// Queue 객체에 지시
Boolean Queue_add(Queue* _this, Element anElement){ // 큐의 rear에
item을 삽입
    if( Queue_isFull(_this)) {
        return FALSE;   // Queue Full 처리
    }
    else {
        _this->_rear = (_this->_rear+1) % _this->_capacity;
        _this->_elements[_this->_rear] = anElement;

        return TRUE;
    }
}

Element Queue_remove(Queue* _this){ // 큐의 front에서 Element를 삭제하
고 그 값을 얻는다
    Element removedElement;
    if( Queue_isEmpty(_this)) {
        return '\0';
    }
    else {
        _this->_front = (_this->_front+1) % _this->_capacity;
        removedElement = _this->_elements[_this->_front];
        return removedElement;
    }
}

Element Queue_elementAt(Queue* _this, int aPosition){   //큐의
aPosition 번째 원소를 얻는다.
    if( aPosition >= _this->_capacity ) {
        return '\0';
    }
    else {
        return _this->_elements[_this->_front+aPosition+1];
    }
}
```

2-7) Queue.h

```
//
//  Queue.h
//  CP2_WEEK13_2
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Queue_h
#define Queue_h
#include "Common.h"

typedef struct _Queue Queue;


// Queue 객체의 생성과 소멸
Queue* Queue_new(int givenCapacity);// Queue 객체의 생성
void Queue_delete(Queue* _this);  // Queue  객체 소멸

// Queue 객체의 점검
Boolean Queue_isEmpty(Queue* _this);    // 큐가 empty이면 TRUE,아니면
FALSE
Boolean Queue_isFull(Queue* _this); // 큐가 full 이면 TRUE를, 아니면,
FALSE
int Queue_size(Queue* _this);   // 큐의 길이, 즉 큐가 가지고 있는 원소의 개
수

// Queue 객체에 지시
Boolean Queue_add(Queue* _this, Element anElement); // 큐의 rear에
item을 삽입
Element Queue_remove(Queue* _this); // 큐의 front에서 Element를 삭제하
고 그 값을 얻는다
Element Queue_elementAt(Queue* _this, int aPosition);   //큐의
aPosition 번째 원소를 얻는다.

#endif /* Queue_h */
```

2-8) Common.h

```
//
//  Common.h
//  CP2_WEEK13_2
//
//  Created by stu2017s10 on 2017. 6. 7..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//
```

```c
#ifndef Common_h
#define Common_h

#include <math.h>
#include <stdlib.h>
#define DEFAULT_QUEUE_CAPACITY  8

#define NewVector(TYPE,SIZE) (TYPE*)malloc(sizeof(TYPE)*SIZE)
#define NewObject(TYPE) (TYPE*)malloc(sizeof(TYPE))
#define isDigit(CHAR) (('0' <= CHAR) && (CHAR <= '9'))
#define isAlpha(CHAR) ( (('a' <= CHAR) && (CHAR <= 'z')) || (('A'
<= CHAR) && (CHAR <= 'Z')))
typedef enum {FALSE,TRUE} Boolean;  // FALSE와 TRUE 값을 갖는 Boolean
선언
typedef int Element;

#endif /* Common_h */
```

3. 전체 설명

3-1. 연결리스트로 구현한 큐

1) main에서 appController 객체를 생성하고, AppController_run(appController)함수를 통해 프로그램을 실행시킨다.

2) AppController_run(appController) 에서는 AppController_initCharCounts() 함수가 입력된 문자의 개수, 무시된 문자의 개수, 삽입된 문자의 개수를 0으로 초기화 시킨다.

3) 그 다음 AppIO_in_nextInputChar(_this->_appIO) 함수로 문자들을 연속적으로 받은 후 inputChar에 저장하고, AppController_countInput(_this) 함수로 입력받은 문자들의 개수를 센다.

4) 입력된 문자가 Esc가 아니면 계속 문자를 입력받고 문자에 따른 함수를 수행한다.

5) 알파벳이 입력되면 AppController_add(_this, inputChar) 함수로 큐에 알파벳을 삽입한다.

6) 숫자가 입력되면 AppController_removeN(_this, inputChar) 함수로 큐에 삽입된 원소를 삭제한다.

7) -가 입력되면 AppController_remove1(_this) 함수로 큐에 삽입된 원소를 삭제한다.

8) #이 입력되면 AppController_showSize(_this) 함수로 큐의 크기를 출력한다.

9) /가 입력되면 AppController_showAllFromFront(_this) 함수로 front부터 큐에 있는 모든 원소를 차례로 출력한다.

10) ^가 입력되면 AppController_showFront(_this) 함수로 front에 있는 원소를 출력한다.

11) 그 외의 문자가 입력되면 AppController_ignore(_this) 함수로 그 문자를 무시하고, 무시된 문자의 개수를 1개 증가시킨다.

12) Esc가 입력되면 지금까지 수행된 결과(입력된 문자의 개수, 정상적으로 처리된 문자의 개수, 무시된 문자의 개수, 삽입된 문자의 개수) 를 출력한다.

3-2. 배열로 구현한 큐

1) main에서 appController 객체를 생성하고, AppController_run(appController)함수를 통해 프로그램을 실행시킨다.

2) AppController_run(appController) 에서는 AppController_initCharCounts() 함수가 입력된 문자의 개수, 무시된 문자의 개수, 삽입된 문자의 개수를 0으로 초기화 시킨다.

3) 그 다음 AppIO_in_nextInputChar(_this->_appIO) 함수로 문자들을 연속적으로 받은 후 inputChar에 저장하고, AppController_countInput(_this) 함수로 입력받은 문자들의 개수를 센다.

4) 입력된 문자가 Esc가 아니면 계속 문자를 입력받고 문자에 따른 함수를 수행한다.

5) 알파벳이 입력되면 AppController_add(_this, inputChar) 함수로 큐에 알파벳을 삽입한다.

6) 숫자가 입력되면 AppController_removeN(_this, inputChar) 함수로 큐에 삽입된 원소를 삭제한다.

7) -가 입력되면 AppController_remove1(_this) 함수로 큐에 삽입된 원소를 삭제한다.

8) #이 입력되면 AppController_showSize(_this) 함수로 큐의 크기를 출력한다.

9) /가 입력되면 AppController_showAllFromFront(_this) 함수로 front부터 큐에 있는 모든 원소를 차례로 출력한다.

10) ^가 입력되면 AppController_showFront(_this) 함수로 front에 있는 원소를 출력한다.

11) 그 외의 문자가 입력되면 AppController_ignore(_this) 함수로 그 문자를 무시하고, 무시된 문자의 개수를 1개 증가시킨다.

12) Esc가 입력되면 지금까지 수행된 결과(입력된 문자의 개수, 정상적으로 처리된 문자의 개수, 무시된 문자의 개수, 삽입된 문자의 개수) 를 출력한다.

4. 실행 결과

4-1. 연결리스트로 구현한 큐

```
>>> 프로그램을 시작합니다 <<<
>>> 문자를 입력하세오 : a

[Add] 삽입된 원소는 'a' 입니다.
>>> 문자를 입력하세오 : b

[Add] 삽입된 원소는 'b' 입니다.
>>> 문자를 입력하세오 : c

[Add] 삽입된 원소는 'c' 입니다.
>>> 문자를 입력하세오 : d

[Add] 삽입된 원소는 'd' 입니다.
>>> 문자를 입력하세오 : 1

[Remove1] 삭제된 원소는 'a' 입니다.
>>> 문자를 입력하세오 : -

[Remove1] 삭제된 원소는 'b' 입니다.
>>> 문자를 입력하세오 : #

[Size] 현재 큐의 크기는 2 입니다.
>>> 문자를 입력하세오 : /
<Front>c d <Rear>
>>> 문자를 입력하세오 : ^

[Front] Front 원소는 'c' 입니다.
>>> 문자를 입력하세오 : &

[Ignore] 의미 없는 문자가 입력되었습니다.
>>> 문자를 입력하세오 :

[Remove1] 삭제된 원소는 'c' 입니다.
>>> 입력된 문자는 11 개 입니다.
>>> 정상적으로 처리된 문자는 10 개 입니다.
>>> 무시된 문자는 1 개 입니다.
>>> 삽입된 문자는 4 개 입니다.
<<< 프로그램을 종료합니다 >>>
Program ended with exit code: 0
```

4-2. 배열로 구현한 큐

```
>>> 프로그램을 시작합니다 <<<
>>> 문자를 입력하세요 : a

[Add] 삽입된 원소는 'a' 입니다.
>>> 문자를 입력하세요 : b

[Add] 삽입된 원소는 'b' 입니다.
>>> 문자를 입력하세요 : c

[Add] 삽입된 원소는 'c' 입니다.
>>> 문자를 입력하세요 : d

[Add] 삽입된 원소는 'd' 입니다.
>>> 문자를 입력하세요 : 1

[Remove1] 삭제된 원소는 'a' 입니다.
>>> 문자를 입력하세요 : -

[Remove1] 삭제된 원소는 'b' 입니다.
>>> 문자를 입력하세요 : #

[Size] 현재 큐의 크기는 2 입니다.
>>> 문자를 입력하세요 : ^

[Front] Front 원소는 'c' 입니다.
>>> 문자를 입력하세요 : /
<Front>c d <Rear>
>>> 문자를 입력하세요 : &

[Ignore] 의미 없는 문자가 입력되었습니다.
>>> 문자를 입력하세요 :

[Remove1] 삭제된 원소는 'c' 입니다.
>>> 입력된 문자는 11 개 입니다.
>>> 정상적으로 처리된 문자는 10 개 입니다.
>>> 무시된 문자는 1 개 입니다.
>>> 삽입된 문자는 4 개 입니다.
<<< 프로그램을 종료합니다 >>>
Program ended with exit code: 0
```