컴퓨터 프로그래밍 2
-hw09-

학번 : 201602038
제출일 : 2017.5.10.
이름 : 이 미 진

1. 함수 설명

1-1. AppIO

1-1) AppIO* AppIO_new() : 객체 생성
1-2) void AppIO_delete(AppIO* _this) : 객체 소멸
1-3) char AppIO_in_charDirectlyFromKeyboard(AppIO* _this) : 문자를 입력 받음
1-4) void AppIO_out_stackIsFullAgainstPush(AppIO* _this, char anElementForPush) : 스택이 꽉 찼을때 삽입할 경우 삽입이 불가능 하다는 메시지 출력
1-5) void AppIO_out_pushedElement(AppIO* _this, char anElement) : 삽입된 원소 출력
1-6) void AppIO_out_stackIsEmptyAgainstPop1(AppIO* _this) : pop할때 스택이 비었을 경우 스택에 삭제할 원소가 없다는 메시지 출력
1-7) void AppIO_out_poppedElementByPop1(AppIO* _this,char anElement) : pop할 때 삭제된 원소가 무엇인지 출력
1-8) void AppIO_out_beginpops(AppIO* _this, int numberOfElements) : 삭제할 원소를 보여줌
1-9) void AppIO_out_endPops(AppIO* _this) : 삭제 종료 메시지 출력
1-10) void AppIO_out_stackIsEmptyAgainstPops(AppIO* _this) : pop할 때 스택이 비었을 경우 스택에 더 이상 삭제할 원소가 없다는 메시지 출력
1-11) void AppIO_out_topElement(AppIO* _this, char anElement) : top에 있는 원소 출력
1-12) void AppIO_out_noTopElement(AppIO* _this) : top에 원소가 없을 경우 스택이 비어있다는 메시지 출력
1-13) void AppIO_out_bottomOfStack(AppIO* _this) : <Bottom of Stack> 출력
1-14) void AppIO_out_element(AppIO* _this, char anElement) : 원소 출력
1-15) void AppIO_out_topOfStack(AppIO* _this) : <Top of Stack> 출력
1-16) void AppIO_out_newLine(AppIO* _this) : 개행
1-17) void AppIO_out_stackSize(AppIO* _this, int stackSize) : 스택의 크기를 출력
1-18) void AppIO_out_ignoredChar(AppIO* _this) : 의미 없는 문자가 입력되었을 때 메시지 출력
1-19) void AppIO_out_endInput(AppIO* _this) : 입력 종료 및 스택의 모든 원소 삭제
1-20) void AppIO_out_poppedElementByEndInput(AppIO* _this, char anElement) : 삭제된 원소가 무엇인지 출력
1-21) void AppIO_out_numberOfInputChars(AppIO* _this, int numberOfChars); // 입력된 문자의 개수 출력
1-22) void AppIO_out_numberOfNomarllyProcessedChars(AppIO* _this, int numberOfChars); // 정상적으로 처리된 문자의 개수 출력
1-23) void AppIO_out_numberOfIgnoredChars(AppIO* _this, int numberOfChars);   // 무시된 문자의 개수 출력
1-24) void AppIO_out_numberOfPushedChars(AppIO* _this, int numberOfChars);   // 스택에 넣은 문자의 개수 출력
1-25) void AppIO_out_endProgram(AppIO* _this);   // 프로그램 종료 메시지

1-2. AppController

2-1) AppController* AppController_new() : 객체 생성
2-2) void AppController_delete(AppController* _this) : 객체 소멸
2-3) void AppController_run(AppController* _this) : 프로그램 실행
2-4) void AppController_push(AppController* _this, char anElement) : 스택에 원소를 넣음
2-5) void AppController_pops(AppController* _this, int numberOfElements) : 스택에서 원소를 꺼내고 삭제
2-6) void AppController_pop1(AppController* _this) : 스택에서 원소를 꺼내고 삭제
2-7) void AppController_showSize(AppController* _this) :스택의 크기를 보여줌
2-8) void AppController_showAllFromBottom(AppController* _this) : 스택의 아래에서부터 원소를 보여줌
2-9) void AppController_showAllFromTop(AppController* _this) : 스택의 위에서부터 원소를 보여줌
2-10) void AppController_showTopElement(AppController* _this) : 스택의 top에 있는 원소를 보여줌
2-11) void AppController_ignore(AppController* _this) : 무시되는 원소를 검사하고 개수 증가
2-12) void AppController_endInput(AppController* _this) : 입력 종료
2-13) void AppController_endProgram(AppController* _this) :프로그램 종료
2-14) void AppController_initCountingChars(AppController* _this) : 각 문자의 개수를 초기화
2-15) void AppController_countInputChars(AppController* _this): 입력된 원소 개수 count
2-16) void AppController_countIgnoredChars(AppController* _this): 무시된 원소 개수 count
2-17) void AppController_countPushedChars(AppController* _this): 스택에 저장된 원소 개수 count
2-18) int AppController_numberOfInputChars(AppController* _this): 입력된 원소 개수를 얻음
2-19) int AppController_numberOfIgnoredChars(AppController* _this): 무시된 원소 개수를 얻음
2-20) int AppController_numberOfNormallyProcessedChars(AppController* _this): 정상적으로 처리된 원소 개수를 얻음
2-21) int AppController_numberOfPushedChars(AppController* _this): 스택에 저장된 원소의 개수를 얻음

1-3. Stack

3-1) Stack* Stack_new(): 스택 객체 생성
3-2) void Stack_delete(Stack* _this): 스택에 있는 원소와 스택 소멸
3-3) void Stack_push(Stack* _this, Element anElement) : 스택에 원소를 넣음
3-4) Boolean Stack_isEmpty(Stack* _this) : 스택이 비었는지 검사
3-5) Boolean Stack_isFull(Stack* _this) : 스택이 꽉 찼는지 검사
3-6) Element Stack_pop(Stack* _this) : 스택에서 원소를 꺼내고 삭제
3-7) Element Stack_elementAt(Stack* _this, int aPosition) :스택 리스트의 aPosition 번째 원소를 얻는다.
3-8) Element Stack_topElement(Stack* _this) : 스택의 top에 있는 원소를 리턴
3-9) int Stack_size(Stack* _this) : 스택의 사이즈를 리턴

2. 전체 코드

2-1. main.c

```
//
//  main.c
//  CP2_WEEK9
//
//  Created by stu2017s10 on 2017. 5. 2..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#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. AppController.c

```
//
//  AppController.c
//  CP2_WEEK9
//
//  Created by stu2017s10 on 2017. 5. 2..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "AppController.h"
#include "Common.h"
#include "AppIO.h"
#include "Stack.h"

#define Esc 27

// 비공개 함수
// 문자 별로 해야 할 일들
void AppController_push(AppController* _this, char anElement);  // 스택에 원소를 넣음
void AppController_pops(AppController* _this, int numberOfElements);    // 스택에서 원소를 꺼내고 삭제
void AppController_pop1(AppController* _this);  // 스택에서 원소를 꺼내고 삭제
void AppController_showSize(AppController* _this);  // 스택의 크기를 보여줌
void AppController_showAllFromBottom(AppController* _this); // 스택의 아래에서부터 원소를 보여줌
void AppController_showAllFromTop(AppController* _this);    // 스택의 위에서부터 원소를 보여줌
void AppController_showTopElement(AppController* _this);    // 스택의 top에 있는 원소를 보여줌
void AppController_ignore(AppController* _this);    // 무시되는 원소를 검사하고 개수 증가
void AppController_endInput(AppController* _this);  // 입력 종료
void AppController_endProgram(AppController* _this);    // 프로그램 종료

// 각 행위 실행 횟수 세기
void AppController_initCountingChars(AppController* _this);    // 각 문자의 개수를 초기화
void AppController_countInputChars(AppController* _this);   // 입력된 원소 개수 count
void AppController_countIgnoredChars(AppController* _this); // 무시된 원소 개수 count
void AppController_countPushedChars(AppController* _this);  // 스택에 저장된 원소 개수 count

// 각 실행 횟수 얻기
```

```
int AppController_numberOfInputChars(AppController* _this); // 입력된 원소 개수를 얻음
int AppController_numberOfIgnoredChars(AppController* _this);   // 무시된 원소 개수를 얻음
int AppController_numberOfNormallyProcessedChars(AppController* _this); // 정상적으로 처리된 원소 개수를 얻음
int AppController_numberOfPushedChars(AppController* _this);    // 스택에 저장된 원소의 개수를 얻음

struct _AppController {
    AppIO* _appIO;
    Stack* _stack;
    int _inputChars;    // 입력된 문자의 개수
    int _ignoredChars;  // 무시된 문자의 개수
    int _pushedChars;   // 스택에 넣은 문자의 개수
};

AppController* AppController_new() {    // 객체 생성
    AppController* _this;
    _this = NewObject(AppController);
    _this->_stack = Stack_new();
    _this->_appIO = AppIO_new();

    return _this;
}

void AppController_delete(AppController* _this) {   // 객체 소멸
    AppIO_delete(_this->_appIO);
    Stack_delete(_this->_stack);

    free(_this);
}

void AppController_push(AppController* _this, char anElement){  // 스택에 원소를 넣음
    if( Stack_isFull (_this->_stack)) {
        AppIO_out_stackIsFullAgainstPush(_this->_appIO, anElement);
    }
    else {
        Stack_push(_this->_stack, anElement);
        AppController_countPushedChars(_this);
        AppIO_out_pushedElement(_this->_appIO, anElement);
    }
}

void AppController_pops(AppController* _this, int numberOfElements) {   // 스택에서 원소를 꺼내고 삭제
    AppIO_out_beginpops(_this->_appIO, numberOfElements);
```

```
    for( int i=0; (i<numberOfElements) && (!Stack_isEmpty(_this->_stack)); i++ ) {
        char poppedChar = Stack_pop(_this->_stack);
        AppIO_out_poppedElementByPop1(_this->_appIO, poppedChar);
    }
    if(Stack_isEmpty(_this->_stack)) {
        AppIO_out_stackIsEmptyAgainstPops(_this->_appIO);
    }
    else {
        AppIO_out_endPops(_this->_appIO);
    }
}

void AppController_pop1(AppController* _this) {   // 스택에서 원소를 꺼내고 삭제
    if( Stack_isEmpty(_this->_stack)) {
        AppIO_out_stackIsEmptyAgainstPop1(_this->_appIO);
    }
    else {
        char poppedChar = Stack_pop(_this->_stack);
        AppIO_out_poppedElementByPop1(_this->_appIO, poppedChar);
    }
}

void AppController_showSize(AppController* _this) {  // 스택의 크기를 보여줌
    int size = Stack_size(_this->_stack);
    AppIO_out_stackSize(_this->_appIO, size);
}

void AppController_showAllFromBottom(AppController* _this) { // 스택의 아래에서부터 원소를 보여줌
    // App 사용자는 스택의 내용에 직접 접근 불가

    char stackElement;
    AppIO_out_bottomOfStack(_this->_appIO);
    for( int i=0; i<Stack_size(_this->_stack); i++ ) {
        stackElement = Stack_elementAt(_this->_stack, i);
        AppIO_out_element(_this->_appIO, stackElement);
    }
    AppIO_out_topOfStack(_this->_appIO);
    AppIO_out_newLine(_this->_appIO);
}

void AppController_showAllFromTop(AppController* _this) {    // 스택의 위에서부터 원소를 보여줌
    char stackElement;
    AppIO_out_topOfStack(_this->_appIO);
    for( int i=0; i<Stack_size(_this->_stack); i++ ) {
        stackElement = Stack_elementAt(_this->_stack, i);
```

```
        AppIO_out_element(_this->_appIO, stackElement);
    }
    AppIO_out_bottomOfStack(_this->_appIO);
    AppIO_out_newLine(_this->_appIO);
}

void AppController_showTopElement(AppController* _this) {    // 스
택의 top에 있는 원소를 보여줌
    char stackElement;
    stackElement = Stack_topElement(_this->_stack);
    AppIO_out_topElement(_this->_appIO, stackElement);
}

void AppController_ignore(AppController* _this) {   // 무시되는 원소를
검사하고 개수 증가
    AppIO_out_ignoredChar(_this->_appIO);
    AppController_countIgnoredChars(_this);
}

void AppController_endInput(AppController* _this) {  // 입력 종료
    AppIO_out_endInput(_this->_appIO);
    char stackElement;
    for(int i = Stack_size(_this->_stack)-1; i>=0; i--){
        stackElement = Stack_elementAt(_this->_stack, i);
        AppIO_out_poppedElementByEndInput(_this->_appIO,
stackElement);
    }
}
void AppController_endProgram(AppController* _this) {    // 프로그램
종료
    AppIO_out_endProgram(_this->_appIO);
}

void AppController_initCountingChars(AppController* _this) {    //
각 문자의 개수를 초기화
    _this->_inputChars = 0;
    _this->_ignoredChars = 0;
    _this->_pushedChars = 0;
}

void AppController_countInputChars(AppController* _this) {   // 입
력된 원소 개수 count
    _this->_inputChars++;
}

void AppController_countIgnoredChars(AppController* _this) { // 무
시된 원소 개수 count
    _this->_ignoredChars++;
}
```

```
void AppController_countPushedChars(AppController* _this) {  // 스
택에 저장된 원소 개수 count
    _this->_pushedChars++;
}

// 통계 정보 얻기
int AppController_numberOfInputChars(AppController* _this) { // 입
력된 원소 개수를 얻음
    return (_this->_inputChars);
}
int AppController_numberOfIgnoredChars(AppController* _this)
{   // 무시된 원소 개수를 얻음
    return (_this->_ignoredChars);
}
int AppController_numberOfNormallyProcessedChars(AppController*
_this) { // 정상적으로 처리된 원소 개수를 얻음
    return (_this->_inputChars - _this->_ignoredChars);
}
int AppController_numberOfPushedChars(AppController* _this)
{    // 스택에 저장된 원소의 개수를 얻음
    return (_this->_pushedChars);
}

// 통계 정보 출력
void AppController_showStatistics(AppController* _this) {
    AppIO_out_newLine(_this->_appIO);

    AppIO_out_numberOfInputChars(_this->_appIO,
AppController_numberOfInputChars(_this));
    AppIO_out_numberOfNormallyProcessedChars(_this->_appIO,
AppController_numberOfNormallyProcessedChars(_this));
    AppIO_out_numberOfIgnoredChars(_this->_appIO,
AppController_numberOfIgnoredChars(_this));
    AppIO_out_numberOfPushedChars(_this->_appIO,
AppController_numberOfPushedChars(_this));
}

void AppController_run(AppController* _this){   // 프로그램 실행
    char c;
    AppController_initCountingChars(_this);
    c = AppIO_in_charDirectlyFromKeyboard(_this->_appIO);

    while(c != Esc){
        AppController_countInputChars(_this);
        if( isAlpha(c) ){
            AppController_push(_this, c);
        }
        else if( isDigit(c)) {
            int digitValue = c - '0';
            AppController_pops(_this, digitValue);
```

```
        }
        else if(c == '-'){
            AppController_pop1(_this);
        }
        else if(c == '#'){
            AppController_showSize(_this);
        }
        else if(c == '/'){
            AppController_showAllFromBottom(_this);
        }
        else if(c == '\\'){
            AppController_showAllFromTop(_this);
        }
        else if(c == '^'){
            AppController_showTopElement(_this);
        }
        else {
            AppController_ignore(_this);
        }
        c = AppIO_in_charDirectlyFromKeyboard(_this->_appIO);
    }
    AppController_endInput(_this);

    AppController_showStatistics(_this);
    AppController_endProgram(_this);
}
```

2-3. AppController.h

```
//  AppController.h
//  CP2_WEEK9
//
//  Created by stu2017s10 on 2017. 5. 2..
//  Copyright © 2017년 stu2017s10. All rights reserved.
#ifndef AppController_h
#define AppController_h
#include <stdio.h>

#define isDigit(CHAR) (('0' <= CHAR) && (CHAR <= '9'))
#define isAlpha(CHAR) ((('A' <= CHAR) && (CHAR <= 'Z')) || (('a'
<= CHAR) && (CHAR <= 'z')))
typedef struct _AppController AppController;

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

#endif /* AppController_h */
```

2-4. Common.h

```
//
//  Common.h
//  CP2_WEEK9
//
//  Created by stu2017s10 on 2017. 5. 2..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Common_h
#define Common_h

#include <stdlib.h>

#define NewVector(TYPE,SIZE) (TYPE*)malloc(sizeof(TYPE)*SIZE)
#define NewObject(TYPE) (TYPE*)malloc(sizeof(TYPE))
typedef enum {FALSE,TRUE} Boolean;  // FALSE와 TRUE 값을 갖는 Boolean 선언

#endif /* Common_h */
```

2-5. AppIO.c

```
//
//  AppIO.c
//  CP2_WEEK9
//
//  Created by stu2017s10 on 2017. 5. 2..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "AppIO.h"
#include "Common.h"
#include "getcharDirectlyFromKeyboard.h"


struct _AppIO {

};

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

    return _this;
}

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

```
}

char AppIO_in_charDirectlyFromKeyboard(AppIO* _this) {  // 문자를 입
력 받음
    printf("문자를 입력하시오 : ");
    char charDirectlyFromKeyboard = getcharDirectlyFromKeyboard();
    AppIO_out_newLine(_this);

    return charDirectlyFromKeyboard;
}

void AppIO_out_stackIsFullAgainstPush(AppIO* _this, char
anElementForPush) { // 스택이 꽉 찼을때 삽입할 경우 삽입이 불가능 하다는 메시지
출력
    printf(" [Push: Full] 스택이 꽉 차서 원소 \'%c\' 는 삽입이 불가능합니다.
\n", anElementForPush);
}

void AppIO_out_pushedElement(AppIO* _this, char anElement) { // 삽
입된 원소 출력
    printf("[Push] 삽입이 된 원소는 \'%c\' 입니다.\n" ,anElement);
}

void AppIO_out_stackIsEmptyAgainstPop1(AppIO* _this) {   // pop할때
스택이 비었을 경우 스택에 삭제할 원소가 없다는 메시지 출력
    printf("[Pop1: Empty] 스택에 삭제할 원소가 없습니다.\n");
}

void AppIO_out_poppedElementByPop1(AppIO* _this,char anElement) {
// pop할 때 삭제된 원소가 무엇인지 출력
    printf("[Pop1] 삭제된 원소는 \'%c\' 입니다.\n" ,anElement);
}

void AppIO_out_beginpops(AppIO* _this, int numberOfElements)
{   // 삭제할 원소를 보여줌
    printf("[Pops] %d 개의 원소를 삭제하려고 합니다.\n",
numberOfElements);
}

void AppIO_out_endPops(AppIO* _this) {   // 삭제 종료 메시지 출력
    printf("[Pops] 삭제를 종료합니다.\n");
}

void AppIO_out_stackIsEmptyAgainstPops(AppIO* _this) {   //  pop할
때 스택이 비었을 경우 스택에 더 이상 삭제할 원소가 없다는 메시지 출력
    printf("[Pops: Empty] 스택에 더 이상 삭제할 원소가 없습니다.\n");
}
```

```c
void AppIO_out_topElement(AppIO* _this, char anElement) {    // top에 있는 원소 출력
    printf("[Top] Top 원소는 \'%c\' 입니다.\n" ,anElement);
}

void AppIO_out_noTopElement(AppIO* _this) {  // top에 원소가 없을 경우 스택이 비어있다는 메시지 출력
    printf("[Top: Empt] 현재 스택은 비어 있습니다.\n");
}

void AppIO_out_bottomOfStack(AppIO* _this) { // <Bottom of Stack> 출력
    printf("<Bottom of Stack> ");
}

void AppIO_out_element(AppIO* _this, char anElement) {   // 원소 출력
    printf("%c ",anElement);
}

void AppIO_out_topOfStack(AppIO* _this) {    // <Top of Stack> 출력
    printf("<Top of Stack>");
}

void AppIO_out_newLine(AppIO* _this) {  // 개행
    printf("\n");
}

void AppIO_out_stackSize(AppIO* _this, int stackSize) {  // 스택의 크기를 출력
    printf("[Size] 현재 스택의 크기는 %d 입니다. \n" , stackSize);
}

void AppIO_out_ignoredChar(AppIO* _this) {   // 의미 없는 문자가 입력되었을 때 메시지 출력
    printf("[Ignore] 의미 없는 문자가 입력되었습니다.\n");
}

void AppIO_out_endInput(AppIO* _this) {  // 입력 종료 및 스택의 모든 원소 삭제
    printf("[End Input] 입력을 종료하며, 스택의 모든 원소를 삭제합니다.\n");
}

void AppIO_out_poppedElementByEndInput(AppIO* _this, char anElement) {   // 삭제된 원소가 무엇인지 출력
    printf(">End Input : 삭제된 원소는 \'%c\' 입니다.\n" ,anElement);
}
```

```
void AppIO_out_numberOfInputChars(AppIO* _this, int numberOfChars)
{ // 입력된 문자의 개수 출력
    printf(">>> 입력된 문자는 모두 %d 개 입니다.\n" ,numberOfChars);
}

void AppIO_out_numberOfNormallyProcessedChars(AppIO* _this, int
numberOfChars) {    // 정상적으로 처리된 문자의 개수 출력
    printf(">>> 정상적으로 처리된 문자는 %d 개 입니다.\n" ,numberOfChars);
}

void AppIO_out_numberOfIgnoredChars(AppIO* _this, int
numberOfChars) {   // 무시된 문자의 개수 출력
    printf(">>> 무시된 문자는 %d 개 입니다.\n" ,numberOfChars);
}

void AppIO_out_numberOfPushedChars(AppIO* _this, int
numberOfChars) {    // 스택에 넣은 문자의 개수 출력
    printf(">>> 스택에 넣은 문자는 %d 개 입니다.\n" ,numberOfChars);
}

void AppIO_out_endProgram(AppIO* _this) {   // 프로그램 종료 메시지 출력
    AppIO_out_newLine(_this);
    printf("> 프로그램을 종료합니다.\n");
}
```

2-6. AppIO.h

```
//
//  AppIO.h
//  CP2_WEEK9
//
//  Created by stu2017s10 on 2017. 5. 2..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppIO_h
#define AppIO_h

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

typedef struct _AppIO   AppIO;

AppIO* AppIO_new(); // 객체 생성
void AppIO_delete(AppIO* _this);    // 객체 소멸
char AppIO_in_charDirectlyFromKeyboard(AppIO* _this);   // 문자를 입
력 받음
```

```
void AppIO_out_stackIsFullAgainstPush(AppIO* _this, char
anElementForPush); // 스택이 꽉 찼을때 삽입할 경우 삽입이 불가능 하다는 메시지
출력
void AppIO_out_pushedElement(AppIO* _this, char anElement); // 삽입
된 원소 출력
void AppIO_out_stackIsEmptyAgainstPop1(AppIO* _this);   // pop할때
스택이 비었을 경우 스택에 삭제할 원소가 없다는 메시지 출력
void AppIO_out_poppedElementByPop1(AppIO* _this,char anElement);
// pop할 때 삭제된 원소가 무엇인지 출력
void AppIO_out_beginpops(AppIO* _this, int numberOfElements);   //
삭제할 원소를 보여줌
void AppIO_out_endPops(AppIO* _this);   // 삭제 종료 메시지 출력
void AppIO_out_stackIsEmptyAgainstPops(AppIO* _this);   //  pop할
때 스택이 비었을 경우 스택에 더 이상 삭제할 원소가 없다는 메시지 출력
void AppIO_out_topElement(AppIO* _this, char anElement);    //
top에 있는 원소 출력
void AppIO_out_noTopElement(AppIO* _this);  // top에 원소가 없을 경우
스택이 비어있다는 메시지 출력
void AppIO_out_bottomOfStack(AppIO* _this); // <Bottom of Stack> 출
력
void AppIO_out_element(AppIO* _this, char anElement);   // 원소 출력
void AppIO_out_topOfStack(AppIO* _this);    // <Top of Stack> 출력
void AppIO_out_newLine(AppIO* _this);   // 개행
void AppIO_out_stackSize(AppIO* _this, int stackSize);  // 스택의 크
기를 출력
void AppIO_out_ignoredChar(AppIO* _this);   // 의미 없는 문자가 입력되었
을 때 메시지 출력
void AppIO_out_endInput(AppIO* _this);  // 입력 종료 및 스택의 모든 원소
삭제
void AppIO_out_poppedElementByEndInput(AppIO* _this, char
anElement);   // 삭제된 원소가 무엇인지 출력

void AppIO_out_numberOfInputChars(AppIO* _this, int
numberOfChars); // 입력된 문자의 개수 출력
void AppIO_out_numberOfNormallyProcessedChars(AppIO* _this, int
numberOfChars); // 정상적으로 처리된 문자의 개수 출력
void AppIO_out_numberOfIgnoredChars(AppIO* _this, int
numberOfChars);   // 무시된 문자의 개수 출력
void AppIO_out_numberOfPushedChars(AppIO* _this, int
numberOfChars);    // 스택에 넣은 문자의 개수 출력
void AppIO_out_endProgram(AppIO* _this);    // 프로그램 종료 메시지 출력

#endif /* AppIO_h */
```

2-7. Stack.c

```
//
//  Stack.c
//  CP2_WEEK9
//
//  Created by stu2017s10 on 2017. 5. 2..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Stack.h"
#include "Common.h"
#define MAX_STACK_SIZE 4

struct _Stack {
    int _top;
    Element* _elements; // 배열을 동적으로 할당
};

Stack* Stack_new() {    // 스택 객체 생성
    Stack* _this;
    _this = NewObject(Stack);
    _this->_elements = NewVector(Element, MAX_STACK_SIZE);
    _this->_top = -1;

    return _this;
}

void Stack_delete(Stack* _this) {    // 스택에 있는 원소와 스택 소멸
    free(_this->_elements);
    free(_this);
}

void Stack_push(Stack* _this, Element anElement) {  // 스택에 원소를
넣음
    if(!Stack_isFull(_this)){
        _this->_top++;
        _this->_elements[_this->_top] = anElement;
    }
    // stack 이 empty이면 push를 무시
}

Boolean Stack_isEmpty(Stack* _this) {   // 스택이 비었는지 검사
    return ((_this->_top) < 0 );
}
Boolean Stack_isFull(Stack* _this) {    // 스택이 꽉 찼는지 검사
    return ( (_this->_top) == (MAX_STACK_SIZE-1) ) ;
}

Element Stack_pop(Stack* _this) {   // 스택에서 원소를 꺼내고 삭제
    // stack은 empty가 아니라고 가정
```

```
    Element poppedElement;
    poppedElement = _this->_elements[_this->_top];
    _this->_top--;
    return poppedElement;
}


Element Stack_elementAt(Stack* _this, int aPosition) {   //스택 리스트의 aPosition 번째 원소를 얻는다.
    return (_this->_elements[aPosition]);

}
Element Stack_topElement(Stack* _this) {    // 스택의 top에 있는 원소를 리턴
    return (_this->_elements[_this->_top]);
}
int Stack_size(Stack* _this) {  // 스택의 사이즈를 리턴
    return (_this->_top+1);
}
```

2-8. Stack.h

```
//
//  Stack.h
//  CP2_WEEK9
//
//  Created by stu2017s10 on 2017. 5. 2..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Stack_h
#define Stack_h

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

typedef struct _Stack Stack;
typedef char Element;


Stack* Stack_new();    // 스택 객체 생성
void Stack_delete(Stack* _this);    // 스택에 있는 원소와 스택 소멸
void Stack_push(Stack* _this, Element anElement);  // 스택에 원소를 넣음
Boolean Stack_isEmpty(Stack* _this);   // 스택이 비었는지 검사
Boolean Stack_isFull(Stack* _this);    // 스택이 꽉 찼는지 검사
Element Stack_pop(Stack* _this);   // 스택에서 원소를 꺼내고 삭제
Element Stack_elementAt(Stack* _this, int aPosition);   //스택 리스트의 aPosition 번째 원소를 얻는다.
```

```
Element Stack_topElement(Stack* _this);    // 스택의 top에 있는 원소를 리턴
int Stack_size(Stack* _this);  // 스택의 사이즈를 리턴

#endif /* Stack_h */
```

3. 종합 설명

1) main에서 AppController_new() 함수를 통해 appController 객체를 생성한다.
2) AppController_run(appController) 함수가 프로그램을 실행한다.
3) AppController_run()에서는 AppController_initCountingChars(_this) 함수가 입력된 문자의 개수,무시된 문자의 개수, 스택에 넣은 문자의 개수를 0으로 초기화하고 AppIO_in_charDirectlyFromKeyboard(_this->_appIO) 함수를 통해 문자를 입력받는다. 그리고 입력된 문자는 c에 저장된다.
4) c가 Esc가 아닐때 AppController_countInputChars(_this) 함수로 입력 받은 문자의 개수를 세고 c가 알파벳일때 , 숫자일 때, 특수 문자일 때의 조건을 따로 두어 함수들을 반복한다.
5) c가 알파벳이면 AppController_push(_this, c) 함수가 스택에 입력받은 문자를 삽입한다.
6) c가 숫자이면 AppController_pops(_this, digitValue) 함수가 원소들을 꺼낸다.
7) c가 '-' 이면 AppController_pop1(_this) 함수가 원소들을 꺼낸다.
8) c가 '#' 이면 AppController_showSize(_this) 함수가 스택의 크기를 보여준다.
9) c가 '/' 이면 AppController_showAllFromBottom(_this)함수가 스택의 bottom에서부터 원소들을 보여준다.
10) c가 '\\'이면 AppController_showAllFromTop(_this) 함수가 스택의 top에서부터 원소들을 보여준다.
11) c가 '^'이면 AppController_showTopElement(_this) 함수가 스택의 top에 있는 원소를 보여준다.
12) 이 외의 것이 입력되면 AppController_ignore(_this) 함수를 통해 입력된 문자를 무시하고, 무시된 원소의 개수를 1 증가시킨다.
13) 조건문을 다 수행하고 다시 입력을 받은 후, Esc가 입력되면 AppController_endInput(_this) 함수로 입력을 종료하고, AppController_showStatistics(_this) 함수로 통계 정보를 출력한다. AppController_endProgram(_this) 함수로 프로그램 종료 메시지를 출력한다.
14)main에서는 AppController_run(appController) 함수가 끝나면 AppController_delete(appController) 함수를 통해 객체를 소멸시킨다.

4. 실행 결과

```
문자를 입력하시오 : A
[Push] 삽입이 된 원소는 'A' 입니다.
문자를 입력하시오 : x
[Push] 삽입이 된 원소는 'x' 입니다.
문자를 입력하시오 : #
[Size] 현재 스택의 크기는 2 입니다.
문자를 입력하시오 : h
[Push] 삽입이 된 원소는 'h' 입니다.
문자를 입력하시오 : /
<Bottom of Stack> A x h <Top of Stack>
문자를 입력하시오 : W
[Push] 삽입이 된 원소는 'W' 입니다.
문자를 입력하시오 : Z
 [Push: Full] 스택이 꽉 차서 원소 'Z' 는 삽입이 불가능
합니다.
문자를 입력하시오 : \
<Top of Stack>A x h W <Bottom of Stack>
문자를 입력하시오 : -
[Pop1] 삭제된 원소는 'W' 입니다.
문자를 입력하시오 : ^
[Top] Top 원소는 'h' 입니다.
문자를 입력하시오 : 4
[Pops] 4 개의 원소를 삭제하려고 합니다.
[Pop1] 삭제된 원소는 'h' 입니다.
[Pop1] 삭제된 원소는 'x' 입니다.
[Pop1] 삭제된 원소는 'A' 입니다.
[Pops: Empty] 스택에 더 이상 삭제할 원소가 없습니다.
문자를 입력하시오 : B
[Push] 삽입이 된 원소는 'B' 입니다.
문자를 입력하시오 : =
[Ignore] 의미 없는 문자가 입력되었습니다.
문자를 입력하시오 : e
[Push] 삽입이 된 원소는 'e' 입니다.
문자를 입력하시오 :
[End Input] 입력을 종료하며, 스택의 모든 원소를 삭제합니
다.
>End Input : 삭제된 원소는 'e' 입니다.
>End Input : 삭제된 원소는 'B' 입니다.

>>> 입력된 문자는 모두 15 개 입니다.
>>> 정상적으로 처리된 문자는 14 개 입니다.
>>> 무시된 문자는 1 개 입니다.
>>> 스택에 넣은 문자는 6 개 입니다.

> 프로그램을 종료합니다.
Program ended with exit code: 0
```