컴퓨터 프로그래밍 2
-hw10-

학번 : 201602038
제출일 : 2017.5.18.
이름 : 이 미 진

1. 함수 설명

1-1. 수식 계산: 초기 프로그램

1) AppIO

1-1) AppIO* AppIO_new() : 객체 생성

1-2) void AppIO_delete(AppIO* _this): 객체 소멸

1-3) Boolean AppIO_in_postfixExpression(AppIO* _this,char* anExpression) : 수식을 입력받음 , $를 입력받으면 종료

1-4) void AppIO_out_errorInExpression (AppIO* _this): 수식에 오류가 있을 때 오류 메시지 출력

1-5) void AppIO_out_evaluatedValue(AppIO* _this, int anEvaluatedValue): 계산값 출력

1-6) void AppIO_out_startingMessage(AppIO* _this): 시작 메시지 출력

1-7) void AppIO_out_endingMessage(AppIO* _this): 종료 메시지 출력

1-8) void AppIO_out_newLine(AppIO* _this) : 개행

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

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_reset(Stack* _this): 스택 초기화

3-4) void Stack_push(Stack* _this, Element anElement): 스택에 원소를 넣음

3-5) Boolean Stack_isEmpty(Stack* _this): 스택이 비었는지 검사

3-6) Boolean Stack_isFull(Stack* _this): 스택이 꽉 찼는지 검사

3-7) Element Stack_pop(Stack* _this): 스택에서 원소를 꺼내고 삭제

3-8) Element Stack_elementAt(Stack* _this, int aPosition):스택 리스트의 aPosition 번째 원소를 얻는다.

3-9) Element Stack_topElement(Stack* _this): 스택의 top에 있는 원소를 리턴

3-10) int Stack_size(Stack* _this): 스택의 사이즈를 리턴

4) Postfix

4-1) Postfix* Postfix_new(int givenMaxNumberOfTokens); // 객체 생성

4-2) void Postfix_delete(Postfix* _this); // 객체 소멸

4-3) void Postfix_setExpression(Postfix* _this, char* anExpression); //계산할 postfix 수식인 expression[] 을 postfix 객체에 전달

4-4) void Postfix_showTokenAndStack(Postfix* _this, char currentToken);

4-5) Boolean Postfix_evaluate(Postfix* _this); //현재 객체가 가지고 있는 postfix 수식을 계산하도록 지시

4-6) int Postfix_evaluatedValue(Postfix* _this); // 계산된 결과 값을 Postifx 객체로부터 얻는다.

1-2. 수식 계산: 안정화 된 프로그램

1) AppIO

1-1) AppIO* AppIO_new(): 객체 생성

1-2) void AppIO_delete(AppIO* _this): 객체 소멸

1-3) Boolean AppIO_in_postfixExpression(AppIO* _this,char* anExpression): 수식을 입력받음 , $를 입력받으면 종료

1-4) void AppIO_out_errorInExpression (AppIO* _this): 수식에 오류가 있을 때 오류 메시지 출력

1-5) void AppIO_out_evaluatedValue(AppIO* _this, int anEvaluatedValue): 계산값 출력

1-6) void AppIO_out_startingMessage(AppIO* _this): 시작 메시지 출력

1-7) void AppIO_out_endingMessage(AppIO* _this): 종료 메시지 출력

1-8) void AppIO_out_postfixEvaluationErrorMessage(AppIO* _this, PostfixError aPostfixError): 오류 메시지 출력 -> 초기 프로그램과 달라진 점

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

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_reset(Stack* _this): 스택 초기화

3-4) void Stack_push(Stack* _this, Element anElement): 스택에 원소를 넣음

3-5) Boolean Stack_isEmpty(Stack* _this): 스택이 비었는지 검사

3-6) Boolean Stack_isFull(Stack* _this): 스택이 꽉 찼는지 검사

3-7) Element Stack_pop(Stack* _this): 스택에서 원소를 꺼내고 삭제

3-8) Element Stack_elementAt(Stack* _this, int aPosition): 스택 리스트의 aPosition 번째 원소를 얻는다.

3-9) Element Stack_topElement(Stack* _this): 스택의 top에 있는 원소를 리턴

3-10) int Stack_size(Stack* _this): 스택의 사이즈를 리턴

4) Postfix

4-1) Postfix* Postfix_new(int givenMaxNumberOfTokens): 객체 생성

4-2) void Postfix_delete(Postfix* _this): 객체 소멸

4-3) void Postfix_setExpression(Postfix* _this, char* anExpression): 계산할 postfix 수식인 expression[] 을 postfix 객체에 전달

4-4) void Postfix_showTokenAndStack(Postfix* _this, char currentToken): 스택과 토큰을 보여줌

4-5) PostfixError Postfix_evaluate(Postfix* _this): 현재 객체가 가지고 있는 postfix 수식을 계산하도록 지시

4-6) int Postfix_evaluatedValue(Postfix* _this): 계산된 결과 값을 Postifx 객체로부터 얻는다.

2. 전체 코드

2-1. 초기 프로그램

1-1) main.c

```
//
//  main.c
//  CP2_WEEK10
//
//  Created by stu2017s10 on 2017. 5. 16..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

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

1-2) AppIO.c

```
//
//  AppIO.c
//  CP2_WEEK10
//
//  Created by stu2017s10 on 2017. 5. 16..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "AppIO.h"

void AppIO_out_newLine(AppIO* _this);
void AppIO_out_bottomOfStack(AppIO* _this);

struct _AppIO {

};

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

    return _this;
}
```

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

Boolean AppIO_in_postfixExpression(AppIO* _this,char*
anExpression) {   // 수식을 입력받음 , $를 입력받으면 종료
    printf(">Postfix 수식을 입력하시오 :");
    scanf("%s",anExpression);
    if( anExpression[0] == '$' ) {
        return FALSE;
    }
    return TRUE;
}
void AppIO_out_errorInExpression (AppIO* _this) {   // 수식에 오류가
있을 때 오류 메시지 출력
    printf(">수식에 오류가 있습니다\n");
}

void AppIO_out_evaluatedValue(AppIO* _this, int anEvaluatedValue)
{  // 계산값 출력
    printf("계산값 : %d\n",anEvaluatedValue);
}

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

void AppIO_out_bottomOfStack(AppIO* _this) {
    printf("<Bottom>");
}


void AppIO_out_startingMessage(AppIO* _this) {  // 시작 메시지 출력
    printf("<Postfix 수식을 계산합니다>\n");
}

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

1-3) AppIO.h

```
//
//  AppIO.h
//  CP2_WEEK10
//
//  Created by stu2017s10 on 2017. 5. 16..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppIO_h
#define AppIO_h

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

typedef struct _AppIO AppIO;

AppIO* AppIO_new();    // 객체 생성
void AppIO_delete(AppIO* _this);    // 객체 소멸
Boolean AppIO_in_postfixExpression(AppIO* _this,char*
anExpression);   // 수식을 입력받음 , $를 입력받으면 종료
void AppIO_out_errorInExpression (AppIO* _this);   // 수식에 오류가 있
을 때 오류 메시지 출력
void AppIO_out_evaluatedValue(AppIO* _this, int anEvaluatedValue);
// 계산값 출력
void AppIO_out_startingMessage(AppIO* _this);  // 시작 메시지 출력
void AppIO_out_endingMessage(AppIO* _this); // 종료 메시지 출력

#endif /* AppIO_h */
```

1-4) AppController.c

```
//
//  AppController.c
//  CP2_WEEK10
//
//  Created by stu2017s10 on 2017. 5. 16..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

void AppController_showAllFromBottom(AppController* _this);

struct _AppController {
```

```
    AppIO* _appIO;
    Stack* _stack;
    char _expression[MAX_NUMBER_OF_TOKENS];
    Postfix* _postfix;
};

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_run(AppController* _this){   // 프로그램 실행
    Boolean expressionIsAvailable, noErrorIsInEvaluation;

    AppIO_out_startingMessage(_this->_appIO);
    _this->_postfix = Postfix_new(MAX_NUMBER_OF_TOKENS);
    expressionIsAvailable = AppIO_in_postfixExpression(_this-
>_appIO,_this->_expression);
    while( expressionIsAvailable ) {
        Postfix_setExpression(_this->_postfix,_this->_expression);
        noErrorIsInEvaluation = Postfix_evaluate(_this->_postfix);

        if( ! noErrorIsInEvaluation ) {
            AppIO_out_errorInExpression(_this->_appIO);
        }
        else {
            AppIO_out_evaluatedValue(_this-
>_appIO,Postfix_evaluatedValue(_this->_postfix));
        }
        expressionIsAvailable = AppIO_in_postfixExpression(_this-
>_appIO,_this->_expression);

    }
    Postfix_delete(_this->_postfix);
    AppIO_out_endingMessage(_this->_appIO);
}
```

1-5) AppController.h

```
//
//  AppController.h
//  CP2_WEEK10
//
//  Created by stu2017s10 on 2017. 5. 16..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppController_h
#define AppController_h

#include <stdio.h>

#define MAX_NUMBER_OF_TOKENS    200

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 */
```

1-6) Stack.c

```
//
//  Stack.c
//  CP2_WEEK10
//
//  Created by stu2017s10 on 2017. 5. 16..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Stack.h"
#include "Common.h"
#define MAX_STACK_SIZE 5

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_reset(Stack* _this) {    // 스택 초기화
    _this->_top =-1;
}

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 리스
트의 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);
}
```

1-7) Stack.h

```
//
//  Stack.h
//  CP2_WEEK10
//
//  Created by stu2017s10 on 2017. 5. 16..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Stack_h
#define Stack_h

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

typedef struct _Stack Stack;
typedef int Element;


Stack* Stack_new();     // 스택 객체 생성
void Stack_delete(Stack* _this);     // 스택에 있는 원소와 스택 소멸
void Stack_reset(Stack* _this); // 스택 초기화
void Stack_push(Stack* _this, Element anElement);  // 스택에 원소를 넣
음
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 리스트
의 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 */
```

1-8) Postfix.c

```
//
//  Postfix.c
//  CP2_WEEK10
//
//  Created by stu2017s10 on 2017. 5. 16..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Postfix.h"
#include "Stack.h"
#include <string.h>
```

```
struct _Postfix {
    int _maxNumberOfTokens;
    char* _expression;
    int _evaluatedValue;
    Stack* _operandStack;
};

Postfix* Postfix_new(int givenMaxNumberOfTokens) {  // 객체 생성
    Postfix* _this = NewObject(Postfix);
    _this->_maxNumberOfTokens = givenMaxNumberOfTokens;
    _this->_expression = NewVector(char, givenMaxNumberOfTokens);
    _this->_operandStack = Stack_new(givenMaxNumberOfTokens);

    return _this;
}


void Postfix_delete(Postfix* _this) {    // 객체 소멸
    Stack_delete(_this->_operandStack);
    free(_this->_expression);
    free(_this);
}

void Postfix_setExpression(Postfix* _this, char* anExpression) {
//계산할 postfix 수식인 expression[] 을 postfix 객체에 전달
    strcpy(_this->_expression,anExpression);
}

void Postfix_showTokenAndStack(Postfix* _this, char currentToken)
{

    printf("%c : Stack <Bottom> ",currentToken);
    for( int i=0; i<Stack_size(_this->_operandStack); i++ ) {
        int Stack = Stack_elementAt(_this->_operandStack, i);
        printf("%d ",Stack);
    }

    printf("<Top>\n");
}

Boolean Postfix_evaluate(Postfix* _this){   //현재 객체가 가지고 있는
postfix 수식을 계산하도록 지시
    int operand, operand1, operand2, calculated;
    char currentToken;
    int i = 0;
    Stack_reset(_this->_operandStack);

    while (_this->_expression[i] != '\0') {
        currentToken = _this->_expression[i];
        if( currentToken >= '0' && currentToken <= '9') {
```

```
            operand = (currentToken-'0');
            Stack_push(_this->_operandStack, operand);
        }

        else {
            if( currentToken == '+') {  // 덧셈
                operand2 = Stack_pop(_this->_operandStack);
                operand1 = Stack_pop(_this->_operandStack);
                calculated = operand1 + operand2;
                Stack_push(_this->_operandStack, calculated);
            }
            else if ( currentToken == '-') {    // 뺄셈
                operand2 = Stack_pop(_this->_operandStack);
                operand1 = Stack_pop(_this->_operandStack);
                calculated = operand1 - operand2;
                Stack_push(_this->_operandStack, calculated);
            }
            else if ( currentToken == '*') {    // 곱셈
                operand2 = Stack_pop(_this->_operandStack);
                operand1 = Stack_pop(_this->_operandStack);
                calculated = operand1 * operand2;
                Stack_push(_this->_operandStack, calculated);
            }
            else if( currentToken == '/') { // 나눗셈
                operand2 = Stack_pop(_this->_operandStack);
                operand1 = Stack_pop(_this->_operandStack);
                calculated = operand1 / operand2;
                Stack_push(_this->_operandStack, calculated);
            }
            else if ( currentToken == '%' ) {   // 나머지 연산
                operand2 = Stack_pop(_this->_operandStack);
                operand1 = Stack_pop(_this->_operandStack);
                calculated = operand1 % operand2;
                Stack_push(_this->_operandStack, calculated);
            }

        }
        Postfix_showTokenAndStack(_this,currentToken);
        i++;
    }// end of while
    _this->_evaluatedValue = Stack_pop(_this->_operandStack);
    return TRUE;
}

int Postfix_evaluatedValue(Postfix* _this){ // 계산된 결과 값을
Postifx 객체로부터 얻는다.
    return (_this->_evaluatedValue);
}
```

1-9) Postfix.h

```
//
//  Postfix.h
//  CP2_WEEK10
//
//  Created by stu2017s10 on 2017. 5. 16..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Postfix_h
#define Postfix_h

#include <stdio.h>
#include "Common.h"
typedef struct _Postfix Postfix;

Postfix* Postfix_new(int givenMaxNumberOfTokens);  // 객체 생성
void Postfix_delete(Postfix* _this);    // 객체 소멸
void Postfix_setExpression(Postfix* _this, char* anExpression); //
계산할 postfix 수식인 expression[] 을 postfix 객체에 전달
void Postfix_showTokenAndStack(Postfix* _this, char currentToken);
Boolean Postfix_evaluate(Postfix* _this);   //현재 객체가 가지고 있는
postfix 수식을 계산하도록 지시
int Postfix_evaluatedValue(Postfix* _this); // 계산된 결과 값을
Postifx 객체로부터 얻는다.

#endif /* Postfix_h */
```

2-10) Common.h

```
//
//  Common.h
//  CP2_WEEK10
//
//  Created by stu2017s10 on 2017. 5. 16..
//  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 */
```

2-2. 안정화 된 프로그램

1-1) main.c

```
//
//  main.c
//  CP2_WEEK10_2
//
//  Created by stu2017s10 on 2017. 5. 16..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

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

2-2) AppIO.c

```
//
//  AppIO.c
//  CP2_WEEK10_2
//
//  Created by stu2017s10 on 2017. 5. 17..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

void AppIO_out_newLine(AppIO* _this);
void AppIO_out_bottomOfStack(AppIO* _this);

struct _AppIO {

};

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

    return _this;
}
```

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

Boolean AppIO_in_postfixExpression(AppIO* _this,char*
anExpression) {   // 수식을 입력받음 , $를 입력받으면 종료
    printf(">Postfix 수식을 입력하시오 :");
    scanf("%s",anExpression);
    if( anExpression[0] == '$' ) {
        return FALSE;
    }
    return TRUE;
}
void AppIO_out_errorInExpression (AppIO* _this) {   // 수식에 오류가
있을 때 오류 메시지 출력
    printf(">수식에 오류가 있습니다\n");
}

void AppIO_out_evaluatedValue(AppIO* _this, int anEvaluatedValue)
{ // 계산값 출력
    printf("계산값 : %d\n",anEvaluatedValue);
}

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

void AppIO_out_bottomOfStack(AppIO* _this) {
    printf("<Bottom>");
}


void AppIO_out_startingMessage(AppIO* _this) {  // 시작 메시지 출력
    printf("<Postfix 수식을 계산합니다>\n");
}

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

void AppIO_out_postfixEvaluationErrorMessage(AppIO* _this,
PostfixError aPostfixError) { // 오류 메시지 출력
    if( aPostfixError == PostfixError_ExpressionTooLong) {
        printf(ErrorMsg_ExpressionTooLong); //[오류] 수식이 너무 길어 처
리가 불가능합니다
        AppIO_out_newLine(_this);
    }
    else if( aPostfixError == PostfixError_OperandsTooMany ) {
```

```
        printf(ErrorMsg_OperandsTooMany);   // [오류] 연산자에 비해 연산
값의 수가 많습니다
        AppIO_out_newLine(_this);
    }
    else if( aPostfixError == PostfixError_OperandsTooFew ) {
        printf(ErrorMsg_OperandsTooFew);    // [오류] 연산자에 비해 연산
값의 수가 적습니다
        AppIO_out_newLine(_this);
    }
    else if( aPostfixError == PostfixError_UnknownOperator ) {
        printf(ErrorMsg_UndefinedOperator);    // [오류] 수식에 알 수
없는 연산자가 있습니다
        AppIO_out_newLine(_this);
    }
    else if( aPostfixError == PostfixError_DivideByZero ) {
        printf(ErrorMsg_DivideByZero);  // [오류] 나눗셈의 분모가 0 입니
다
        AppIO_out_newLine(_this);
    }
}
```

2-3) AppIO.h

```
//
//  AppIO.h
//  CP2_WEEK10_2
//
//  Created by stu2017s10 on 2017. 5. 17..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppIO_h
#define AppIO_h

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

typedef struct _AppIO AppIO;

AppIO* AppIO_new();     // 객체 생성
void AppIO_delete(AppIO* _this);     // 객체 소멸
Boolean AppIO_in_postfixExpression(AppIO* _this,char*
anExpression);   // 수식을 입력받음 , $를 입력받으면 종료
void AppIO_out_errorInExpression (AppIO* _this);   // 수식에 오류가 있
을 때 오류 메시지 출력
void AppIO_out_evaluatedValue(AppIO* _this, int anEvaluatedValue);
// 계산값 출력
void AppIO_out_startingMessage(AppIO* _this);  // 시작 메시지 출력
```

```
void AppIO_out_endingMessage(AppIO* _this); // 종료 메시지 출력
void AppIO_out_postfixEvaluationErrorMessage(AppIO* _this,
PostfixError aPostfixError); // 오류 메시지 출력

#endif /* AppIO_h */
```

2-4) AppController.c

```
//
//  AppController.c
//  CP2_WEEK10_2
//
//  Created by stu2017s10 on 2017. 5. 17..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

void AppController_showAllFromBottom(AppController* _this);

struct _AppController {
    AppIO* _appIO;
    Stack* _stack;
    char _expression[MAX_NUMBER_OF_TOKENS];
    Postfix* _postfix;
};

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_run(AppController* _this){   // 프로그램 실행
    Boolean expressionIsAvailable;
    PostfixError evaluationError;
```

```
    AppIO_out_startingMessage(_this->_appIO);   // 시작 메시지 출력
    _this->_postfix = Postfix_new(MAX_NUMBER_OF_TOKENS);
    expressionIsAvailable = AppIO_in_postfixExpression(_this-
>_appIO,_this->_expression);
    while( expressionIsAvailable ) {
        Postfix_setExpression(_this->_postfix,_this->_expression);
        evaluationError = Postfix_evaluate(_this->_postfix);

        if( evaluationError == PostfixError_None ) {
           AppIO_out_evaluatedValue(_this-
>_appIO,Postfix_evaluatedValue(_this->_postfix));
        }
        else {
            AppIO_out_postfixEvaluationErrorMessage(_this->_appIO,
evaluationError);
        }
        expressionIsAvailable = AppIO_in_postfixExpression(_this-
>_appIO,_this->_expression);

    }
    Postfix_delete(_this->_postfix);
    AppIO_out_endingMessage(_this->_appIO);
}
```

2-5) AppController.h

```
//
//  AppController.h
//  CP2_WEEK10_2
//
//  Created by stu2017s10 on 2017. 5. 17..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppController_h
#define AppController_h
#include <stdio.h>

#define MAX_NUMBER_OF_TOKENS    200

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 */
```

2-6) Stack.c

```c
//
//  Stack.c
//  CP2_WEEK10_2
//
//  Created by stu2017s10 on 2017. 5. 17..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Stack.h"
#include "Common.h"
#define MAX_STACK_SIZE 5

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

Stack* Stack_new(int givenMaxSize) {    // 스택 객체 생성
    Stack* _this;
    _this = NewObject(Stack);
    _this->_maxSize = givenMaxSize;
    _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_reset(Stack* _this) {    // 스택 초기화
    _this->_top =-1;
}

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 리스
트의 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);
}
```

2-7) Stack.h

```
//
//  Stack.h
//  CP2_WEEK10_2
//
//  Created by stu2017s10 on 2017. 5. 17..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Stack_h
#define Stack_h
#include <stdio.h>
#include "Common.h"

typedef struct _Stack Stack;
typedef int Element;


Stack* Stack_new();    // 스택 객체 생성
void Stack_delete(Stack* _this);    // 스택에 있는 원소와 스택 소멸
void Stack_reset(Stack* _this); // 스택 초기화
```

```
void Stack_push(Stack* _this, Element anElement);  // 스택에 원소를 넣
음
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 리스트
의 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 */
```

2-8) Postfix.c

```
//
//  Postfix.c
//  CP2_WEEK10_2
//
//  Created by stu2017s10 on 2017. 5. 17..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Postfix.h"
#include "Stack.h"
#include "AppIO.h"
#include "Common.h"
#include "MessageKOR_PostfixError.h"
#include <string.h>


struct _Postfix {
    int _maxNumberOfTokens;
    char* _expression;
    int _evaluatedValue;
    Stack* _operandStack;
};

Postfix* Postfix_new(int givenMaxNumberOfTokens) {  // 객체 생성
    Postfix* _this = NewObject(Postfix);
    _this->_maxNumberOfTokens = givenMaxNumberOfTokens;
    _this->_expression = NewVector(char, givenMaxNumberOfTokens);
    _this->_operandStack = Stack_new(givenMaxNumberOfTokens);

    return _this;
}


void Postfix_delete(Postfix* _this) {    // 객체 소멸
```

```
    Stack_delete(_this->_operandStack);
    free(_this->_expression);
    free(_this);
}


void Postfix_setExpression(Postfix* _this, char* anExpression) {
    //계산할 postfix 수식인 expression[] 을 postfix 객체에 전달
    strcpy(_this->_expression,anExpression);
}

void Postfix_showTokenAndStack(Postfix* _this, char currentToken)
{
    // 스택과 토큰을 보여줌

    printf("%c : Stack <Bottom> ",currentToken);
    for( int i=0; i<Stack_size(_this->_operandStack); i++ ) {
        int Stack = Stack_elementAt(_this->_operandStack, i);
        printf("%d ",Stack);
    }

    printf("<Top>\n");
}

PostfixError Postfix_evaluate(Postfix* _this){   //현재 객체가 가지고
있는 postfix 수식을 계산하도록 지시
    int operand, operand1, operand2, calculated;
    char currentToken;
    int i = 0;
    Stack_reset(_this->_operandStack);

    while (_this->_expression[i] != '\0') {
        currentToken = _this->_expression[i];
        if( currentToken >= '0' && currentToken <= '9') {
            operand = (currentToken-'0');
            if( Stack_isFull(_this->_operandStack)) {
                return PostfixError_ExpressionTooLong;
            }
            else {
            Stack_push(_this->_operandStack, operand);
            }
        }

        else {
            if( currentToken == '+') {  // 덧셈
                if( Stack_size(_this->_operandStack) >= 2 ) {
                operand2 = Stack_pop(_this->_operandStack);
                operand1 = Stack_pop(_this->_operandStack);
                calculated = operand1 + operand2;
                Stack_push(_this->_operandStack, calculated);
```

```
                // 2개 pop 했으므로, 스택에 하나 push 할 여유는 있으므로
isFull 검사 불필요
            }
            else {
                // [오류] 연산자에 비해 연산값의 수가 많습니다.
                return PostfixError_OperandsTooMany;
            }
        }
        else if ( currentToken == '-') {    // 뺄셈
            if( Stack_size(_this->_operandStack) >= 2 ) {
            operand2 = Stack_pop(_this->_operandStack);
            operand1 = Stack_pop(_this->_operandStack);
            calculated = operand1 - operand2;
            Stack_push(_this->_operandStack, calculated);
        }
            else {
                // [오류] 연산자에 비해 연산값의 수가 많습니다.
                return PostfixError_OperandsTooMany;
            }
        }
        else if ( currentToken == '*') {    // 곱셈
            if( Stack_size(_this->_operandStack) >= 2 ) {
            operand2 = Stack_pop(_this->_operandStack);
            operand1 = Stack_pop(_this->_operandStack);
            calculated = operand1 * operand2;
            Stack_push(_this->_operandStack, calculated);
            }
            else {
                // [오류] 연산자에 비해 연산값의 수가 많습니다.
                return PostfixError_OperandsTooMany;
            }
        }
        else if( currentToken == '/') { // 나눗셈
            if( Stack_size(_this->_operandStack) >= 2 ) {
            operand2 = Stack_pop(_this->_operandStack);
            operand1 = Stack_pop(_this->_operandStack);
                if( operand2 == 0 ) {
                    return PostfixError_DivideByZero;
                }
            calculated = operand1 / operand2;
            Stack_push(_this->_operandStack, calculated);
        }
            else {
                // [오류] 연산자에 비해 연산값의 수가 많습니다.
                return PostfixError_OperandsTooMany;
            }
        }
        else if ( currentToken == '%' ) {   // 나머지 연산
            if( Stack_size(_this->_operandStack) >= 2 ) {
            operand2 = Stack_pop(_this->_operandStack);
```

```
                operand1 = Stack_pop(_this->_operandStack);
                calculated = operand1 % operand2;
                Stack_push(_this->_operandStack, calculated);
            }
                else {

                    // [오류] 연산자에 비해 연산값의 수가 많습니다.
                    return PostfixError_OperandsTooMany;
                }
            }
            else {
                // [오류] 수식에 알 수 없는 연산자가 있습니다.
                return PostfixError_UnknownOperator;
            }

        }
        Postfix_showTokenAndStack(_this,currentToken);
        i++;
    }   // end of while
    if( Stack_size(_this->_operandStack) == 1 ) {
        // the result is on top of stack
        _this->_evaluatedValue = Stack_pop(_this->_operandStack);
    }
    else if ( Stack_size(_this->_operandStack) > 1 ) {
        // [오류] 연산자에 비해 연산값의 수가 적습니다.
        return PostfixError_OperandsTooFew;
    }
    return PostfixError_None;   //  오류 없음. 성공적으로 계산 완료
}

int Postfix_evaluatedValue(Postfix* _this){ // 계산된 결과 값을
Postifx 객체로부터 얻는다.
    return (_this->_evaluatedValue);
}
```

2-9) Postfix.h

```
//
//  Postfix.h
//  CP2_WEEK10_2
//
//  Created by stu2017s10 on 2017. 5. 17..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Postfix_h
#define Postfix_h
```

```
#include <stdio.h>
#include "Common.h"
typedef struct _Postfix Postfix;

Postfix* Postfix_new(int givenMaxNumberOfTokens);  // 객체 생성
void Postfix_delete(Postfix* _this);    // 객체 소멸
void Postfix_setExpression(Postfix* _this, char* anExpression); //
계산할 postfix 수식인 expression[] 을 postfix 객체에 전달
void Postfix_showTokenAndStack(Postfix* _this, char currentToken);
// 스택과 토큰을 보여줌
PostfixError Postfix_evaluate(Postfix* _this);   //현재 객체가 가지고
있는 postfix 수식을 계산하도록 지시
int Postfix_evaluatedValue(Postfix* _this); // 계산된 결과 값을
Postifx 객체로부터 얻는다.

#endif /* Postfix_h */
```

2-10) Common.h

```
//
//  Common.h
//  CP2_WEEK10_2
//
//  Created by stu2017s10 on 2017. 5. 17..
//  Copyright © 2017년 stu2017s10. All rights reserved.

#ifndef Common_h
#define Common_h

#include <stdlib.h>
#include "MessageKOR_PostfixError.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

typedef
enum {
    PostfixError_None,
    PostfixError_ExpressionTooLong,
    PostfixError_OperandsTooMany,
    PostfixError_OperandsTooFew,
    PostfixError_UnknownOperator,
    PostfixError_DivideByZero
}
PostfixError ;


#endif /* Common_h */
```

2-11) MessageKOR_PostfixError.h

```
//
//  MessageKOR_PostfixError.h
//  CP2_WEEK10_2
//
//  Created by stu2017s10 on 2017. 5. 17..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef MessageKOR_PostfixError_h
#define MessageKOR_PostfixError_h

#define ErrorMsg_ExpressionTooLong  "[오류] 수식이 너무 길어 처리가 불가능
합니다.\n"
#define ErrorMsg_OperandsTooMany    "[오류] 연산자에 비해 연산값의 수가 많
습니다.\n"
#define ErrorMsg_OperandsTooFew "[오류] 연산자에 비해 연산값의 수가 적습니
다.\n"
#define ErrorMsg_UndefinedOperator  "[오류]수식에 알 수 없는 연산자가 있습
니다.\n"
#define ErrorMsg_DivideByZero   "[오류] 나눗셈의 분모가 0 입니다.\n"

#endif /* MessageKOR_PostfixError_h */
```

3. 전체 설명

3-1. 초기 프로그램

1) main에서 AppController_new() 함수를 통해 appController 객체를 생성한다.

2) AppController_run(appController) 함수로 프로그램을 실행하고 AppController_run(appController) 에서는 AppIO_out_startingMessage(_this->_appIO) 함수가 시작 메시지를 출력한다.

3) Postfix_new(MAX_NUMBER_OF_TOKENS) 함수로 객체를 생성하는데 여기서 MAX_NUMBER_OF_TOKENS 는 200이다.

4) AppIO_in_postfixExpression(_this->_appIO,_this->_expression) 함수를 통해 수식을 입력 받고, expressionIsAvailable 일 때 계속 반복한다.

5) Postfix_setExpression(_this->_postfix,_this->_expression) 함수를 통해 계산할 postfix 수식인 expression[] 을 postfix 객체에 전달하고, Postfix_evaluate(_this->_postfix) 함수로 현재 객체가 가지고 있는 postfix 수식을 계산하도록 지시한다.

6) 입력 받은 수식에 에러가 있다면 AppIO_out_errorInExpression(_this->_appIO) 함수가 ">수식에 오류가 있습니다" 메시지를 출력한다.

7) 입력 받은 수식에 에러가 없다면 AppIO_out_evaluatedValue(_this->_appIO,Postfix_evaluatedValue(_this->_postfix) 함수가 계산값을 출력한다.

8) 계산값이 출력되면 AppIO_in_postfixExpression(_this->_appIO,_this->_expression) 함수가 다시 수식을 입력 받고, $가 입력된다면 반복을 종료한다.

9) Postfix_delete(_this->_postfix) 함수로 객체를 소멸시킨다.

10) 프로그램이 종료되면서 AppIO_out_endingMessage(_this->_appIO) 함수가 종료 메시지를 출력한다.

3-2. 안정화 된 프로그램

1) main에서 AppController_new() 함수를 통해 appController 객체를 생성한다.

2) AppController_run(appController) 함수로 프로그램을 실행하고 AppController_run(appController) 에서는 AppIO_out_startingMessage(_this->_appIO) 함수가 시작 메시지를 출력한다.

3) Postfix_new(MAX_NUMBER_OF_TOKENS) 함수로 객체를 생성하는데 여기서 MAX_NUMBER_OF_TOKENS 는 200이다.

4) AppIO_in_postfixExpression(_this->_appIO,_this->_expression) 함수를 통해 수식을 입력 받고, expressionIsAvailable 일 때 계속 반복한다.

5) Postfix_setExpression(_this->_postfix,_this->_expression) 함수를 통해 계산할 postfix 수식인 expression[] 을 postfix 객체에 전달하고, Postfix_evaluate(_this->_postfix) 함수로 현재 객체가 가지고 있는 postfix 수식을 계산하도록 지시한다.

6) 만약 입력받은 수식에 에러가 없다면(evaluationError == PostfixError_None) AppIO_out_evaluatedValue(_this->_appIO,Postfix_evaluatedValue(_this->_postfix) 함수가 계산값을 출력한다.

7) 입력받은 수식에 에러가 있다면 AppIO_out_postfixEvaluationErrorMessage(_this->_appIO, evaluationError) 함수를 통해 각 에러에 맞는 메시지를 출력한다.

8) 계산값이 출력되면 AppIO_in_postfixExpression(_this->_appIO,_this->_expression) 함수가 다시 수식을 입력 받고, $가 입력된다면 반복을 종료한다.

9) Postfix_delete(_this->_postfix) 함수로 객체를 소멸시킨다.

10) 프로그램이 종료되면서 AppIO_out_endingMessage(_this->_appIO) 함수가 종료 메시지를 출력한다.

4. 실행 결과

4-1. 초기 프로그램

```
<Postfix 수식을 계산합니다>
>Postfix 수식을 입력하시오 :38-
3 : Stack <Bottom> 3 <Top>
8 : Stack <Bottom> 3 8 <Top>
- : Stack <Bottom> -5 <Top>
계산값 : -5
>Postfix 수식을 입력하시오 :57*8%
5 : Stack <Bottom> 5 <Top>
7 : Stack <Bottom> 5 7 <Top>
* : Stack <Bottom> 35 <Top>
8 : Stack <Bottom> 35 8 <Top>
% : Stack <Bottom> 3 <Top>
계산값 : 3
>Postfix 수식을 입력하시오 :875-/94-12+-*
8 : Stack <Bottom> 8 <Top>
7 : Stack <Bottom> 8 7 <Top>
5 : Stack <Bottom> 8 7 5 <Top>
- : Stack <Bottom> 8 2 <Top>
/ : Stack <Bottom> 4 <Top>
9 : Stack <Bottom> 4 9 <Top>
4 : Stack <Bottom> 4 9 4 <Top>
- : Stack <Bottom> 4 5 <Top>
1 : Stack <Bottom> 4 5 1 <Top>
2 : Stack <Bottom> 4 5 1 2 <Top>
+ : Stack <Bottom> 4 5 3 <Top>
- : Stack <Bottom> 4 2 <Top>
* : Stack <Bottom> 8 <Top>
계산값 : 8
>Postfix 수식을 입력하시오 :82/3-42*+
8 : Stack <Bottom> 8 <Top>
2 : Stack <Bottom> 8 2 <Top>
/ : Stack <Bottom> 4 <Top>
3 : Stack <Bottom> 4 3 <Top>
- : Stack <Bottom> 1 <Top>
4 : Stack <Bottom> 1 4 <Top>
2 : Stack <Bottom> 1 4 2 <Top>
* : Stack <Bottom> 1 8 <Top>
+ : Stack <Bottom> 9 <Top>
계산값 : 9
>Postfix 수식을 입력하시오 :97%78*-253/+*
9 : Stack <Bottom> 9 <Top>
7 : Stack <Bottom> 9 7 <Top>
% : Stack <Bottom> 2 <Top>
7 : Stack <Bottom> 2 7 <Top>
8 : Stack <Bottom> 2 7 8 <Top>
* : Stack <Bottom> 2 56 <Top>
- : Stack <Bottom> -54 <Top>
2 : Stack <Bottom> -54 2 <Top>
5 : Stack <Bottom> -54 2 5 <Top>
3 : Stack <Bottom> -54 2 5 3 <Top>
/ : Stack <Bottom> -54 2 1 <Top>
+ : Stack <Bottom> -54 3 <Top>
* : Stack <Bottom> -162 <Top>
계산값 : -162
>Postfix 수식을 입력하시오 :$
<계산을 종료합니다>
Program ended with exit code: 0
```

4-2. 안정화 된 프로그램

```
<Postfix 수식을 계산합니다>
>Postfix 수식을 입력하시오 :38--
3 : Stack <Bottom> 3 <Top>
8 : Stack <Bottom> 3 8 <Top>
- : Stack <Bottom> -5 <Top>
[오류] 연산자에 비해 연산값의 수가 많습니다.

>Postfix 수식을 입력하시오 :57*8%9
5 : Stack <Bottom> 5 <Top>
7 : Stack <Bottom> 5 7 <Top>
* : Stack <Bottom> 35 <Top>
8 : Stack <Bottom> 35 8 <Top>
% : Stack <Bottom> 3 <Top>
9 : Stack <Bottom> 3 9 <Top>
[오류] 연산자에 비해 연산값의 수가 적습니다.

>Postfix 수식을 입력하시오 :123456+++++
1 : Stack <Bottom> 1 <Top>
2 : Stack <Bottom> 1 2 <Top>
3 : Stack <Bottom> 1 2 3 <Top>
4 : Stack <Bottom> 1 2 3 4 <Top>
5 : Stack <Bottom> 1 2 3 4 5 <Top>
[오류] 수식이 너무 길어 처리가 불가능 합니다.

>Postfix 수식을 입력하시오 :82^
8 : Stack <Bottom> 8 <Top>
2 : Stack <Bottom> 8 2 <Top>
[오류]수식에 알 수 없는 연산자가 있습니다.

>Postfix 수식을 입력하시오 :97%78*-253/+*
9 : Stack <Bottom> 9 <Top>
7 : Stack <Bottom> 9 7 <Top>
% : Stack <Bottom> 2 <Top>
7 : Stack <Bottom> 2 7 <Top>
8 : Stack <Bottom> 2 7 8 <Top>
* : Stack <Bottom> 2 56 <Top>
- : Stack <Bottom> -54 <Top>
2 : Stack <Bottom> -54 2 <Top>
5 : Stack <Bottom> -54 2 5 <Top>
3 : Stack <Bottom> -54 2 5 3 <Top>
/ : Stack <Bottom> -54 2 1 <Top>
+ : Stack <Bottom> -54 3 <Top>
* : Stack <Bottom> -162 <Top>
계산값 : -162
>Postfix 수식을 입력하시오 :$
<계산을 종료합니다>
Program ended with exit code: 0
```