

컴퓨터 프로그래밍 2
-hw06-

학번 : 201602038
제출일 : 2017.4.12.
이름 : 이 미 진

1. 함수설명

1-1) AppIO

- 1) AppIO* AppIO_new() : 객체 생성
- 2) void AppIO_delete(AppIO* _this) : 객체 소멸
- 3) void AppIO_out_msg_startScoreProcessing (AppIO* _this): 성적 처리 시작 메시지 출력
- 4) void AppIO_out_msg_endScoreProcessing (AppIO* _this): 성적 처리 종료 메시지 출력
- 5) void AppIO_out_msg_noStudentFromInput (AppIO* _this): 학생 정보가 입력되지 않았을 때 오류메시지 출력
- 6) void AppIO_out_msg_errorInInputStudentInfo (AppIO* _this): 입력된 성적이 0보다 작거나 100보다 클 때 오류 메시지 출력
- 7) void AppIO_out_msg_invalidScore(AppIO* _this,int aScore): 성적이 0보다 작거나 클 때 오류메시지 출력
- 8) void AppIO_out_averageScore (AppIO* _this, int anAverageScore): 평균 점수 출력
- 9) void AppIO_out_numberOfStudentsAboveAverage (AppIO* _this, int aNumber): 평균 이상인 학생 수 출력
- 10) void AppIO_out_maxScore (AppIO* _this, int aMaxScore): 최고점 출력
- 11) void AppIO_out_minScore (AppIO* _this, int aMinScore):최저점 출력
- 12) void AppIO_out_gradeCountFor (AppIO* _this, char aGrade, int aCount): 학점 별 학생 수 출력
- 13) void AppIO_out_titleForSortedStudentList (AppIO* _this) : 성적순 목록 출력 안내 메시지 출력
- 14) void AppIO_out_studentInfo (AppIO* _this, int aScore, char aGrade) : 점수와 학점 출력
- 15) Boolean AppIO_in_doesContinueToInputNextStudent (AppIO* _this) : 성적을 입력 받음
- 16) int AppIO_in_score (AppIO* _this) : 점수를 입력 받음
- 17) char getcharDirectlyFromKeyboard (void) : keyboard에서 직접 문자 입력 받기

1-2) AppController

- 1) Boolean AppController_inputAndStoreStudents(AppController* _this) : 성적을 입력 받는다.
- 2) AppController* AppController_new(void) : 객체 생성
- 3) void AppController_showStatistics(AppController* _this) : 평균 점수,평균 이상인 학생 수, 최고점, 최저점 출력
- 4) void AppController_showStudentsSortedByScore (AppController* _this) : 학생 정보 정렬
- 5) void AppController_delete(AppController* _this) : 객체 소멸
- 6) void AppController_run(AppController* _this) : 함수들을 실행하고 결과를 보여줌

1-3) Ban

- 1) Ban* Ban_new(void) : 객체 생성자
- 2) Ban* Ban_newWithCapacity(int givenCapacity) : 최대 학생 수 설정 및 초기 학생수 설정
- 3) Boolean Ban_scoreValid(int aScore) : 점수가 0보다 작거나 큰지 검사
- 4) char Ban_scoreToGrade(int aScore) : 점수에 따른 성적 값 return
- 5) void Ban_delete(Ban* _this) : 객체 소멸자

- 6) int Ban_maxSize(Ban* _this): 학급 객체가 가질 수 있는 최대 학생 수를 얻어 알아낸다.
- 7) int Ban_size(Ban* _this) : 학급 객체가 가지고 있는 학생 수를 얻어 알아낸다.
- 8) Boolean Ban_isEmpty(Ban* _this): 현재 가지고 있는 학생이 없으면 TRUE, 없으면 FALSE 를 얻는다.
- 9) Boolean Ban_isFull(Ban* _this) : 현재 학생을 더 이상 저장할 공간이 없으면 TRUE, 아직 여유가 있으면 FALSE를 얻는다.
- 10) Boolean Ban_add(Ban* _this,int aScore) : 학생 정보를 입력 받아 저장하고 정상적으로 성적이 저장되었으면 TRUE, 아니면 즉 저장공간이 모자라면 FALSE를 return 값으로 돌려 받는다.
- 11) int Ban_elementAt(Ban* _this,int anOrder) :Ban에 저장된 원소를 얻어냄
- 12) void Ban_sortStudentsByScore(Ban* _this); : 객체에 저장된 학생들의 성적을 성적 순으로 정렬시킨다.
- 13) void Ban_quickSortRecursively(Ban* _this,int left, int right) : 퀵 정렬
- 14) int Ban_partition(Ban* _this, int left, int right) : 원소들을 특정 값을 기준으로 두 부분으로 나눈다.
- 15) float Ban_averageScore(Ban* _this) : 평균을 계산하여 return 값으로 돌려준다.
- 16) int Ban_maxScore(Ban* _this) : 최고점을 찾아서 return 값으로 돌려준다
- 17) int Ban_minScore(Ban* _this) : 최저점을 찾아서 return 값으로 돌려준다.
- 18) int Ban_sumOfScoreRecursively(Ban* _this,int left, int right) : 성적 합계를 계산하여 return 값으로 돌려준다.
- 19) int Ban_maxScoreRecursively(Ban* _this,int left, int right) : 최고점을 찾아서 return 값으로 돌려준다
- 20) int Ban_minScoreRecursively(Ban* _this,int left, int right) : 최저점을 찾아서 return 값으로 돌려준다.
- 21) int Ban_numberOfStudentsAboveAverage(Ban* _this) : 평균 이상인 학생 수 세기
- 22) GradeCounter* Ban_countGrades(Ban* _this) : 학점 별 학생수를 구함

1-4) GradeCounter

- 1)GradeCounter* GradeCounter_new(void) : 객체를 생성하여 그 소유권을 돌려받는다. 학점을 셀 수 있도록 준비
- 2) void GradeCounter_delete(GradeCounter* _this): 사용이 끝난 객체를 소멸시킨다.
- 3) void GradeCounter_count(GradeCounter* _this, char aGrade) : 주어진 학점을 받아서 해당 학점의 학생수를 증가시키게 함
- 4) int GradeCounter_numberOfA (GradeCounter* _this) : A 학점 학생 수 얻기
- 5) int GradeCounter_numberOfB (GradeCounter* _this) : B 학점 학생 수 얻기
- 6) int GradeCounter_numberOfC (GradeCounter* _this) : C 학점 학생 수 얻기
- 7) int GradeCounter_numberOfD (GradeCounter* _this) : D 학점 학생 수 얻기
- 8) int GradeCounter_numberOfF (GradeCounter* _this) : F 학점 학생 수 얻기

2. 전체 코드

1) main.c

```
//
//  main.c
//  CP2_WEEK6
//
//  Created by stu2017s10 on 2017. 4. 11..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

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

    return 0;
}
```

2) AppIO.c

```
//
//  AppIO.c
//  CP2_WEEK6
//
//  Created by stu2017s10 on 2017. 4. 11..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

struct _AppIO{
};

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

```

}

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

// 출력을 위한 공개 함수
void AppIO_out_msg_startScoreProcessing (AppIO* _this) {    // 성적
처리 시작 메시지 출력
    printf("<<<성적 처리를 시작합니다>>>\n");
}
void AppIO_out_msg_endScoreProcessing (AppIO* _this) {    // 성적 처리
종료 메시지 출력
    printf(">>>프로그램을 종료합니다<<<\n");
}
void AppIO_out_msg_noStudentFromInput (AppIO* _this) {    // 학생 정보
가 입력되지 않았을 때 오류메시지 출력
    printf("[오류] 학생 정보가 전혀 입력되지 않았습니다.\n");
}
void AppIO_out_msg_errorInInputStudentInfo (AppIO* _this) { // 입력
종료 메시지
    printf("입력을 종료합니다.\n");
}
void AppIO_out_msg_invalidScore(AppIO* _this,int aScore) {    // 성적
이 0보다 작거나 클 때 오류메시지 출력
    printf("[오류] 성적이 0보다 작거나 100보다 커서, 정상적인 성적이 아닙니
다.\n");
}

void AppIO_out_averageScore (AppIO* _this, int anAverageScore) {
// 평균 점수 출력
    printf("평균 점수는 %.1d 입니다\n",anAverageScore);
}
void AppIO_out_numberOfStudentsAboveAverage (AppIO* _this, int
aNumber) {    // 평균 이상인 학생의 수 출력
    printf("평균 이상인 학생은 모두 %d 명입니다.\n",aNumber);
}
void AppIO_out_maxScore (AppIO* _this, int aMaxScore) { // 최고점 출
력
    printf("최고점은 %d 점 입니다.\n",aMaxScore);
}
void AppIO_out_minScore (AppIO* _this, int aMinScore) { // 최저점 출
력
    printf("최저점은 %d 점 입니다.\n" ,aMinScore);
}
void AppIO_out_gradeCountFor (AppIO* _this, char aGrade, int
aCount) {    // 학점 별 학생 수 출력
    printf("%c 학점은 %d명 입니다.\n" ,aGrade,aCount);
}
}

```

```

void AppIO_out_titleForSortedStudentList (AppIO* _this) {    // 성적
순 목록 출력 안내 메시지
    printf("학생들의 성적순 목록입니다.\n");
}
void AppIO_out_studentInfo (AppIO* _this, int aScore , char
aGrade) {    // 점수와 학점 출력
    printf("점수 : %d      학점 : %c \n", aScore,aGrade);
}

char getcharDirectlyFromKeyboard (void) {    // keyboard에서 직접 문자
입력 받기
    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, &oldAttr);

    return charFromKeyBoard;
}

// 입력을 위한 공개 함수
//성적을 입력 받음
Boolean AppIO_in_doesContinueToInputNextStudent (AppIO* _this) {
    printf(">>>성적을 입력하려면 'y'를, 입력을 종료하려면 다른 아무 키나 누르시
오.");
    char answer = getcharDirectlyFromKeyboard();
    return ( answer == 'Y' || answer =='y');
}

// 점수를 입력 받음
int AppIO_in_score (AppIO* _this) {
    int score;
    printf(">>>점수를 입력하십시오 : \n");
    scanf("%d",&score);
    return score;
}

```

3) AppIO.h

```
//
// AppIO.h
// CP2_WEEK6
//
// Created by stu2017s10 on 2017. 4. 11..
// Copyright © 2017년 stu2017s10. All rights reserved.
//
// 헤더 파일은 public으로 보면 됨.

#ifndef AppIO_h
#define AppIO_h

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

typedef struct _AppIO AppIO ;

AppIO* AppIO_new(); // 객체 생성
void AppIO_delete(AppIO* _this); // 객체 소멸

// 출력을 위한 공개 함수
void AppIO_out_msg_startScoreProcessing (AppIO* _this); // 성적 처리
시작 메시지 출력
void AppIO_out_msg_endScoreProcessing (AppIO* _this); // 성적 처리
종료 메시지 출력
void AppIO_out_msg_noStudentFromInput (AppIO* _this); // 학생 정보
가 입력되지 않았을 때 오류메시지 출력
void AppIO_out_msg_errorInInputStudentInfo (AppIO* _this); // 입력
된 성적이 0보다 작거나 100보다 클 때 오류 메시지 출력
void AppIO_out_msg_invalidScore(AppIO* _this, int aScore); // 성적
이 0보다 작거나 클 때 오류메시지 출력

void AppIO_out_averageScore (AppIO* _this, int anAverageScore); //
평균 점수 출력
void AppIO_out_numberOfStudentsAboveAverage (AppIO* _this, int
aNumber); // 평균 이상인 학생 수 출력
void AppIO_out_maxScore (AppIO* _this, int aMaxScore); // 최고점 출
력
void AppIO_out_minScore (AppIO* _this, int aMinScore); // 최저점 출
력
void AppIO_out_gradeCountFor (AppIO* _this, char aGrade, int
aCount); // 학점 별 학생 수 출력
void AppIO_out_titleForSortedStudentList (AppIO* _this); //
void AppIO_out_studentInfo (AppIO* _this, int aScore, char
aGrade);
```

```

char getcharDirectlyFromKeyboard (void);    // keyboard에서 직접 문자
입력 받기

// 입력을 위한 공개 함수
Boolean AppIO_in_doesContinueToInputNextStudent (AppIO* _this); //
성적을 입력 받음
int AppIO_in_score (AppIO* _this);    // 점수를 입력 받음

#endif /* AppIO_h */

```

4) AppController.c

```

//
// AppController.c
// CP2_WEEK6
//
// Created by stu2017s10 on 2017. 4. 11..
// Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "AppController.h"
#include "Common.h"

struct _AppController{
    AppIO* _appIO;
    Ban* _ban;
};

AppController* AppController_new(void) {    // 객체 생성
    AppController* _this = NewObject(AppController);
    _this->_appIO = AppIO_new();
    _this->_ban = Ban_newWithCapacity(MAX_NUMBER_OF_STUDENTS);
    return _this;
}

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

// 학생들의 정보를 입력 받아 Ban 객체에 저장한다.
// 저장이 정상적으로 이뤄지지 않으면 더 이상 입력 받지 않음
Boolean AppController_inputAndStoreStudents( AppController* _this)
{
    int score;
    Boolean storingAStudentWasSuccessful = TRUE;
}

```



```

    while (storingAStudentWasSuccessful &&
AppIO_in_doesContinueToInputNextStudent(_this->_appIO)) {
        printf("\n");
        score = AppIO_in_score(_this->_appIO);
        if( Ban_scoreIsValid(score)) {
            storingAStudentWasSuccessful = Ban_add(_this->_ban,
score);
        }
        else {
            AppIO_out_msg_invalidScore(_this->_appIO,score);
        }
    }
    return storingAStudentWasSuccessful;
}

void AppController_showStatistics(AppController* _this) {
    //이 시점에 성적 처리된 결과를 Ban객체가 가지고 있다.
    printf("\n");
    AppIO_out_averageScore(_this->_appIO, Ban_averageScore(_this->_ban));
    AppIO_out_numberOfStudentsAboveAverage(_this->_appIO,
Ban_numberOfStudentsAboveAverage(_this->_ban));
    AppIO_out_maxScore(_this->_appIO, Ban_maxScore(_this->_ban));
    AppIO_out_minScore(_this->_appIO, Ban_minScore(_this->_ban));

    printf("\n");
    //학점 별 학생수는 Ban객체로부터 GradeCounter 객체 형태로 얻음
    GradeCounter* gradeCounter = Ban_countGrades(_this->_ban);
    AppIO_out_gradeCountFor(_this->_appIO, 'A',
GradeCounter_numberOfA(gradeCounter));
    AppIO_out_gradeCountFor(_this->_appIO, 'B',
GradeCounter_numberOfB(gradeCounter));
    AppIO_out_gradeCountFor(_this->_appIO, 'C',
GradeCounter_numberOfC(gradeCounter));
    AppIO_out_gradeCountFor(_this->_appIO, 'D',
GradeCounter_numberOfD(gradeCounter));
    AppIO_out_gradeCountFor(_this->_appIO, 'F',
GradeCounter_numberOfF(gradeCounter));
    GradeCounter_delete(gradeCounter); // 더 이상 필요 없으므로 소멸
}

void AppController_showStudentsSortedByScore (AppController*
_this) {
    printf("\n");
    AppIO_out_titleForSortedStudentList(_this->_appIO);

    int score;
    char grade;
    for( int order = 0; order < Ban_size(_this->_ban); order++ ) {
        score = Ban_elementAt(_this->_ban, order);
        grade = Ban_scoreToGrade(score);
    }
}

```

```

        AppIO_out_studentInfo(_this->_appIO, score, grade);
    }
}

void AppController_run(AppController* _this) {
    AppIO_out_msg_startScoreProcessing(_this->_appIO);

    //성적을 입력받음
    Boolean inputAndStoreWasSuccessful;
    inputAndStoreWasSuccessful =
    AppController_inputAndStoreStudents(_this);

    if( inputAndStoreWasSuccessful ) {
        if( Ban_isEmpty(_this->_ban)) {
            AppIO_out_msg_noStudentFromInput(_this->_appIO);
        }
        else {
            // 평균과 평균 이상인 학생수, 최고점, 최저점 출력
            AppController_showStatistics(_this);
            // 성적순으로 정렬, 입력받은 학생 정보 출력
            Ban_sortStudentsByScore(_this->_ban);
            AppController_showStudentsSortedByScore(_this);
        }
    }
    else {
        AppIO_out_msg_errorInInputStudentInfo(_this->_appIO);
    }
    AppIO_out_msg_endScoreProcessing(_this->_appIO);
}

```

5) AppController.h

```

//
// AppController.h
// CP2_WEEK6
//
// Created by stu2017s10 on 2017. 4. 11..
// Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppController_h
#define AppController_h

#include <stdio.h>
#include "AppIO.h"
#include "Ban.h"
#define MAX_NUMBER_OF_STUDENTS 100

typedef struct _AppController AppController;

```

```

// 학생들의 정보를 입력 받아 Ban 객체에 저장한다.
// 저장이 정상적으로 이뤄지지 않으면 더 이상 입력 받지 않음
Boolean AppController_inputAndStoreStudents( AppController*
_this); // 학생들의 정보를 입력 받아 Ban 객체에 저장한다.
AppController* AppController_new(void); // 객체 생성
void AppController_showStatistics(AppController* _this); // 평균
점수, 평균 이상인 학생 수, 최고점, 최저점 출력
void AppController_showStudentsSortedByScore (AppController*
_this); // 학생 정보 정렬
void AppController_delete(AppController* _this); // 객체 소멸
void AppController_run(AppController* _this);

#endif /* AppController_h */

```

6) Common.h

```

//
// Common.h
// CP2_WEEK6
//
// Created by stu2017s10 on 2017. 4. 11..
// Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Common_h
#define Common_h

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

#define NewObject(TYPE) (TYPE*)malloc(sizeof(TYPE))
#define SWAP(TYPE,X,Y) { TYPE temp=X; X=Y; Y=temp; }
// SWAP() 매크로는 변수 X와 Y의 값을 맞바꿔주는 일을 하는 코드를 생성시켜 줌

#define NewVector(TYPE,SIZE) (TYPE*) malloc(sizeof(TYPE)*SIZE)
// this._elements = NewVector(int, this_maxSize);
typedef enum {FALSE,TRUE} Boolean; // FALSE와 TRUE 값을 갖는 Boolean
선언

#endif /* Common_h */

```

7) Ban.c

```
//
// Ban.c
// CP2_WEEK6
//
// Created by stu2017s10 on 2017. 4. 11..
// Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Ban.h"
#include "Common.h"

struct _Ban {
    int _maxSize;
    int _size;
    int* _elements;
};

Ban* Ban_new(void){ // 객체 생성자
    Ban* _this = NewObject(Ban);

    _this->_maxSize = DEFAULT_CAPACITY; // 최대 학생 수 설정
    _this->_elements = NewVector(int, _this->_maxSize);
    _this->_size = 0; // 객체를 생성한 직후의 학생수는 0명

    return _this;
}

Ban* Ban_newWithCapacity(int givenCapacity) { //최대 학생 수 설정 및
초기 학생수 설정
    Ban* _this = NewObject(Ban);
    _this->_maxSize = givenCapacity; // 최대 학생 수 설정
    _this->_elements = NewVector(int, givenCapacity);
    _this->_size = 0; // 객체를 생성한 직후의 학생수는 0명

    return _this;
}

Boolean Ban_scoreIsValid(int aScore) { // 점수가 0보다 작거나 큰지 검사
    return ( aScore >= 0 && aScore <= 100 );
}

char Ban_scoreToGrade(int aScore) {
    if( aScore >= 90 ) {
        return 'A';
    }
    else if( aScore >= 80 ) {
        return 'B';
    }
    else if( aScore >= 70 ) {
```

```

        return 'C';
    }
    else if( aScore >= 60 ) {
        return 'D';
    }
    else {
        return 'F';
    }
}

void Ban_delete(Ban* _this){    // 객체 소멸자
    free(_this);
}
// Ban객체의 상태 알아보기
int Ban_maxSize(Ban* _this){
    return (_this->_maxSize);
}

int Ban_size(Ban* _this) {
    return (_this->_size);
}

Boolean Ban_isEmpty(Ban* _this) {
    return (_this->_size == 0);
}

Boolean Ban_isFull(Ban* _tihs) {
    return (_tihs->_size >=_tihs->_maxSize);
}

// 학생 개인 정보 입력 받아 저장
Boolean Ban_add(Ban* _this,int aScore) {
    //성적을 배열에 저장
    if( Ban_isFull(_this)) {
        return FALSE;    // 저장 공간 부족
    }
    else {
        // 배열의 맨 마지막 위치에 넣는다.
        _this->_elements[_this->_size] = aScore;
        _this->_size++;
        return TRUE;
    }
}

// Ban에 저장된 원소 얻어내기
int Ban_elementAt(Ban* _this,int anOrder) {
    if( anOrder >= _this->_size) {
        // 주어진 위치에 원소가 존재하지 않는다.
        return -1;
    }
}

```

```

    else {
        // 원소가 정상적으로 존재한다.
        return (_this->_elements[anOrder]);
    }
}

// 학생 정보 정렬(성적 순)
void Ban_sortStudentsByScore(Ban* _this) {
    int size = _this->_size;

    // 정렬할 데이터는 _this->_elements[0] ~ _this->_elements[size-1]
    // 퀵 정렬 실행
    if( size >= 2 ) { // 개수가 2개 이상이면
        //최소값의 위치를 찾는다.
        int minPosition = 0;
        for( int i=1; i<size; i++ ) {
            if(_this->_elements[i] < _this->_elements[minPosition]) {
                minPosition = i;
            }
        }
        // 최소값을 원소 구간의 맨 끝으로 옮긴다.
        SWAP(int, _this->_elements[minPosition], _this->_elements[size-1]);

        // 정렬 시작
        Ban_quickSortRecursively(_this, 0, size-2);
    }
}

// 학생 정보 정렬( 성적 순)
void Ban_quickSortRecursively(Ban* _this, int left, int right) {
    if( left < right ) {
        int mid = Ban_partition(_this, left, right);
        Ban_quickSortRecursively(_this, left, mid-1);
        Ban_quickSortRecursively(_this, mid+1, right);
    }
}

int Ban_partition(Ban* _this, int left, int right) {
    int pivot = left;
    int pivotScore = _this->_elements[pivot];

    right++;
    do {
        do { left++; } while (_this->_elements[left] > pivotScore );
        do { right--; } while (_this->_elements[right] < pivotScore );
        if( left < right ) {

```

```

        SWAP(int, _this->_elements[left], _this->_elements[right]);
    }
    while ( left < right );
    SWAP(int, _this->_elements[pivot], _this->_elements[right]);
    return right;
}

float Ban_averageScore(Ban* _this){ // 평균을 계산하여 return 값으로 돌려준다.
    float sumOfScores = (float) Ban_sumOfScoreRecursively(_this, 0, _this->_size-1);
    float average = sumOfScores / (float) _this->_size;
    return average;
}

int Ban_maxScore(Ban* _this) { // 최고점을 찾아서 return 값으로 돌려준다
    return Ban_maxScoreRecursively(_this, 0, _this->_size-1);
}

int Ban_minScore(Ban* _this){ // 최저점을 찾아서 return 값으로 돌려준다.
    return Ban_minScoreRecursively(_this, 0, _this->_size-1);
}

//재귀함수로 구현
int Ban_sumOfScoreRecursively(Ban* _this, int left, int right) {
    // 성적 합계를 계산하여 return 값으로 돌려준다.
    // 크기를 (N-1)로 줄이는 재귀함수
    if( left > right ) {
        return 0;
    }
    else {
        return (_this->_elements[left] + Ban_sumOfScoreRecursively(_this, left+1, right));
    }
}

int Ban_maxScoreRecursively(Ban* _this, int left, int right) {
    // 최고점을 찾아서 return 값으로 돌려준다
    // 두개의 구간으로 나누는 재귀함수
    int maxOfLeft;
    int maxOfRight;
    int middle;

    if( left == right ) { // left 와 right 같을때
        return _this->_elements[left]; // _elements의 left값 반환
    }
    else {
        middle = (left + right )/2; // middle은 left+right를 2로 나눈 값

```

```

        maxOfLeft = Ban_maxScoreRecursively(_this, left, middle);
        maxOfRight = Ban_maxScoreRecursively(_this, middle+1,
right);

        if( maxOfLeft >= maxOfRight ) { // maxOfLeft가 maxOfRight보
다 크거나 같을 때
            return maxOfLeft;
        }
        else {
            return maxOfRight;
        }
    }
}

int Ban_minScoreRecursively(Ban* _this, int left, int right) {
    // 최저점을 찾아서 return 값으로 돌려준다.
    // 크기를 (N-1)로 줄이는 재귀함수
    int minOfLeft;
    int minOfRight;
    int middle;

    if( left == right ) { // left 와 right 같을때
        return _this->elements[left]; // _elements의 left값 반환
    }
    else {
        middle = (left + right)/2; // middle은 left+right를 2로 나눈
값
        minOfLeft = Ban_minScoreRecursively(_this, left, middle);
        minOfRight = Ban_minScoreRecursively(_this, middle+1,
right);

        if( minOfLeft <= minOfRight ) { // maxOfLeft가 maxOfRight보
다 작거나 같을 때
            return minOfLeft;
        }
        else {
            return minOfRight;
        }
    }
}

// 평균 이상인 학생 수 세기
int Ban_numberOfStudentsAboveAverage(Ban* _this) {
    float average = Ban_averageScore(_this);
    int numberOfStudentsAboveAverage = 0;

    for( int i=0; i<_this->_size; i++ ) {
        if( ((float)_this->elements[i]) >= average) {

```



```

        numberOfStudentsAboveAverage ++;
    }
    return numberOfStudentsAboveAverage;
}

GradeCounter* Ban_countGrades(Ban* _this) { //학점 별 학생수를 구함
    char currentGrade;
    GradeCounter* gradeCounter = GradeCounter_new();

    for( int i=0; i<_this->_size; i++ ) {
        currentGrade = Ban_scoreToGrade(_this->_elements[i]);
        GradeCounter_count(gradeCounter, currentGrade);
    }
    return gradeCounter;
}

```

8) Ban.h

```

//
// Ban.h
// CP2_WEEK6
//
// Created by stu2017s10 on 2017. 4. 11..
// Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Ban_h
#define Ban_h

#define DEFAULT_CAPACITY    100
#include <stdio.h>
#include "Common.h"
#include "GradeCounter.h"

typedef struct _Ban Ban;

Ban* Ban_new(void); // 객체 생성자
Ban* Ban_newWithCapacity(int givenCapacity); // 최대 학생 수 설정 및
초기 학생수 설정
Boolean Ban_scoreIsValid(int aScore); // 점수가 0보다 작거나 큰지 검
사
char Ban_scoreToGrade(int aScore);
void Ban_delete(Ban* _this); // 객체 소멸자

// Ban객체의 상태 알아보기
int Ban_maxSize(Ban* _this); // 학급 객체가 가질 수 있는 최대 학생 수를
얻어 알아낸다.

```

```

int Ban_size(Ban* _this);    // 학급 객체가 가지고 있는 학생 수를 얻어 알아낸다.
Boolean Ban_isEmpty(Ban* _this);    // 현재 가지고 있는 학생이 없으면 TRUE, 없으면 FALSE 를 얻는다.
Boolean Ban_isFull(Ban* _this); // 현재 학생을 더 이상 저장할 공간이 없으면 TRUE, 아직 여유가 있으면 FALSE를 얻는다.

// 학생 개인 정보 입력 받아 저장
// 정상적으로 성적이 저장되었으면 TRUE, 아니면 즉 저장공간이 모자라면 FALSE를 return 값으로 돌려 받는다.
Boolean Ban_add(Ban* _this, int aScore);

// Ban에 저장된 원소 얻어내기
int Ban_elementAt(Ban* _this, int anOrder);

// 학생 정보 정렬(성적 순)
void Ban_sortStudentsByScore(Ban* _this);    // 객체에 저장된 학생들의 성적을 성적 순으로 정렬시킨다.
void Ban_quickSortRecursively(Ban* _this, int left, int right); // 퀵 정렬
int Ban_partition(Ban* _this, int left, int right); // 원소들을 특정 값을 기준으로 두 부분으로 나눈다.

float Ban_averageScore(Ban* _this); // 평균을 계산하여 return 값으로 돌려준다.
int Ban_maxScore(Ban* _this);    // 최고점을 찾아서 return 값으로 돌려준다
int Ban_minScore(Ban* _this);    // 최저점을 찾아서 return 값으로 돌려준다.

//재귀함수로 구현
int Ban_sumOfScoreRecursively(Ban* _this, int left, int right);    // 성적 합계를 계산하여 return 값으로 돌려준다.
int Ban_maxScoreRecursively(Ban* _this, int left, int right);    // 최고점을 찾아서 return 값으로 돌려준다
int Ban_minScoreRecursively(Ban* _this, int left, int right);    // 최저점을 찾아서 return 값으로 돌려준다.

// 평균 이상인 학생 수 세기
int Ban_numberOfStudentsAboveAverage(Ban* _this);

GradeCounter* Ban_countGrades(Ban* _this);    // 학점 별 학생수를 구함

#endif /* Ban_h */

```

9) GradeCounter.c

```

//
// GradeCounter.c
// CP2_WEEK6
//

```

```

// Created by stu2017s10 on 2017. 4. 11..
// Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "GradeCounter.h"
#include "Common.h"

struct _GradeCounter {
    int _numberOfA;
    int _numberOfB;
    int _numberOfC;
    int _numberOfD;
    int _numberOfF;
};

// 객체를 생성하여 그 소유권을 돌려받는다. 학점을 셀 수 있도록 준비
GradeCounter* GradeCounter_new(void) {
    GradeCounter* _this = NewObject(GradeCounter);

    _this->_numberOfA = 0;
    _this->_numberOfB = 0;
    _this->_numberOfC = 0;
    _this->_numberOfD = 0;
    _this->_numberOfF = 0;

    return _this;
}

// 사용이 끝난 객체를 소멸시킨다.
void GradeCounter_delete(GradeCounter* _this) {
    free(_this);
}

// 주어진 학점을 받아서 해당 학점의 학생수를 증가시키게 함
void GradeCounter_count(GradeCounter* _this, char aGrade) {
    switch (aGrade) {
        case 'A':
            _this->_numberOfA++;
            break;
        case 'B':
            _this->_numberOfB++;
            break;
        case 'C':
            _this->_numberOfC++;
            break;
        case 'D':
            _this->_numberOfD++;
            break;
        default:
            _this->_numberOfF++;
    }
}

```

```

}

// 학점 별 학생 수 얻기
int GradeCounter_numberOfA (GradeCounter* _this){    // A 학점 학생 수
얻기
    return _this->_numberOfA;
}
int GradeCounter_numberOfB (GradeCounter* _this){    // B 학점 학생 수
얻기
    return _this->_numberOfB;
}
int GradeCounter_numberOfC (GradeCounter* _this){    // C 학점 학생 수
얻기
    return _this->_numberOfC;
}
int GradeCounter_numberOfD (GradeCounter* _this){    // D 학점 학생 수
얻기
    return _this->_numberOfD;
}
int GradeCounter_numberOfF (GradeCounter* _this){    // F 학점 학생 수
얻기
    return _this->_numberOfF;
}
}

```

10) GradeCounter.h

```

//
// GradeCounter.h
// CP2_WEEK6
//
// Created by stu2017s10 on 2017. 4. 11..
// Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef GradeCounter_h
#define GradeCounter_h

#include <stdio.h>

typedef struct _GradeCounter GradeCounter;

// 객체를 생성하여 그 소유권을 돌려받는다. 학점을 셀 수 있도록 준비
GradeCounter* GradeCounter_new(void);

// 사용이 끝난 객체를 소멸시킨다.
void GradeCounter_delete(GradeCounter* _this);

// 주어진 학점을 받아서 해당 학점의 학생수를 증가시키게 함
void GradeCounter_count(GradeCounter* _this, char aGrade);

```

```
// 학점 별 학생 수 얻기
int GradeCounter_numberOfA (GradeCounter* _this);    // A 학점 학생 수
얻기
int GradeCounter_numberOfB (GradeCounter* _this);    // B 학점 학생 수
얻기
int GradeCounter_numberOfC (GradeCounter* _this);    // C 학점 학생 수
얻기
int GradeCounter_numberOfD (GradeCounter* _this);    // D 학점 학생 수
얻기
int GradeCounter_numberOfF (GradeCounter* _this);    // F 학점 학생 수
얻기

#endif /* GradeCounter_h */
```

3. 전체 설명

- 1) main에서 AppController_new()함수를 통해 appController 객체를 생성해준다.
- 2) AppController_run()함수가 성적 처리 프로그램을 실행한다.
- 3) AppController_run()에서는AppIO_out_msg_startScoreProcessing() 함수가 성적 처리 시작 메시지를 출력하고, AppController_inputAndStoreStudents() 함수가 성적을 입력 받는다.
- 4) 입력된 성적은 inputAndStoreWasSuccessful에 저장되고 Ban_isEmpty()함수를 통해 입력된 성적들의 크기를 검사한다. 만약 크기가 0이라면 AppIO_out_msg_noStudentFromInput() 함수가 “[오류] 학생 정보가 전혀 입력되지 않았습니다.” 메시지를 출력한다.
- 5) 크기가 0이 아니라면 AppController_showStatistics() 함수가 성적들의 평균, 평균 이상인 학생의 수, 최고점, 최저점을 출력한다.
- 6) Ban_sortStudentsByScore() 함수가 성적순으로 정렬을 하고 AppController_showStudentsSortedByScore() 함수가 정렬된 학생들의 정보를 출력한다.
- 7) Ban_sortStudentsByScore() 에서는 퀵 정렬을 사용해 성적의 개수가 2개 이상이면 최소값의 위치를 찾고 최소값을 원소 구간의 맨 끝으로 옮긴다. 그 후 Ban_quickSortRecursively() 함수로 정렬을 시작한다.
- 8) y가 아닌 다른 값이 입력되면 AppIO_out_msg_errorInInputStudentInfo() 함수가 입력을 종료하는 메시지를 출력한다.
- 9) 그 후 AppIO_out_msg_endScoreProcessing() 함수가 성적 처리 종료 메시지를 출력한다.

4. 실행 결과

```
<<<성적 처리를 시작합니다>>>
>>>성적을 입력하려면 'y'를, 입력을 종료하려면 다른 아무 키나 누르시오.y
>>>점수를 입력하십시오 :
82
>>>성적을 입력하려면 'y'를, 입력을 종료하려면 다른 아무 키나 누르시오.y
>>>점수를 입력하십시오 :
45
>>>성적을 입력하려면 'y'를, 입력을 종료하려면 다른 아무 키나 누르시오.y
>>>점수를 입력하십시오 :
93
>>>성적을 입력하려면 'y'를, 입력을 종료하려면 다른 아무 키나 누르시오.y
>>>점수를 입력하십시오 :
102
[오류] 성적이 0보다 작거나 100보다 커서, 정상적인 성적이 아닙니다.
>>>성적을 입력하려면 'y'를, 입력을 종료하려면 다른 아무 키나 누르시오.y
>>>점수를 입력하십시오 :
66
>>>성적을 입력하려면 'y'를, 입력을 종료하려면 다른 아무 키나 누르시오.y
>>>점수를 입력하십시오 :
-1
[오류] 성적이 0보다 작거나 100보다 커서, 정상적인 성적이 아닙니다.
>>>성적을 입력하려면 'y'를, 입력을 종료하려면 다른 아무 키나 누르시오.y
>>>점수를 입력하십시오 :
52
>>>성적을 입력하려면 'y'를, 입력을 종료하려면 다른 아무 키나 누르시오.n
평균 점수는 67 입니다
평균 이상인 학생은 모두 2 명입니다.
최고점은 93 점 입니다.
최저점은 45 점 입니다.

A 학점은 1명 입니다.
B 학점은 1명 입니다.
C 학점은 0명 입니다.
D 학점은 1명 입니다.
F 학점은 2명 입니다.

학생들의 성적순 목록입니다.
점수 : 93      학점 : A
점수 : 82      학점 : B
점수 : 66      학점 : D
점수 : 52      학점 : F
점수 : 45      학점 : F
>>>프로그램을 종료합니다<<<
Program ended with exit code: 0
```