컴퓨터 프로그래밍 2
-hw04-

학번 : 201602038
이름 : 이 미 진

1. 함수 설명

1-1. QuadEquation

1) AppIO

1-1) void AppIO_out_msg_startSolvlingQuadEquation(); :<<<이차방정식 풀이>>> 메시지 출력

1-2) void AppIO_out_msg_endSolvingQuadEquation(); : <<<이차방정식 풀이를 종료합니다>>> 메시지 출력

1-3) void AppIO_out_msg_secondOrderTermCoefficientIsZero(void) ; : 이차항의 계수가 0임을 알리는 메시지 출력

1-4) void AppIO_out_msg_determinantIsNegative (float aDeterminant) ; : 판별식의 값이 음수임을 알리는 메시지 출력

1-5) Boolean AppIO_in_solvingIsRequested (void) ; : 이차방정식 풀이를 할 것인지 종료할 것인지 묻는 메시지 출력

1-6) void AppIO_in_quadEquation (float* p_c0, float* p_c1, float* p_c2) : 계수를 입력받아 돌려준다.

1-7) void AppIO_out_quadEquation (float c0, float c1, float c2) ; : 주어진 방정식을 출력한다.

1-8) void AppIO_out_determinant(float aDeterminant); : 판별식의 값을 보여준다.

1-9) void AppIO_out_roots (float root1, float root2) ; : 방정식의 해를 보여준다.

2) QuadEquationProblem

2-1) void QuadEquationProblem_setEquation (QuadEquationProblem* _this, QuadEquation anEquation) ; : 객체에게 정보를 제공하는 함수

2-2) Boolean QuadEquationProblem_secondOrderTermCoefficientIsZero(QuadEquationProblem* _this); : 이차항이 0인지 아닌지 상태를 검사하는 함수

2-3) Boolean QuadEquationProblem_determinantIsNegative (QuadEquationProblem*_this); : 판별식이 0인지 아닌지 상태를 검사하는 함수

2-4) void QuadEquationProblem_solve (QuadEquationProblem* _this) ; : 방정식의 해를 구하는 함수

2-5) QuadEquation QuadEquationProblem_equation(QuadEquationProblem* _this); : 소유하고 있는 정보로부터 객체의 상태를 얻는다.

2-6) float QuadEquationProblem_determinant(QuadEquationProblem* _this); :주어진 이차방정식 문제 _this에 대해 판별식을 계산하는 함수

2-7) Solution QuadEquationProblem_solution(QuadEquationProblem* _this); : 소유하고 있는 정보로부터 객체의 상태를 얻는다.

1-2. MagicSquare

1) AppIO

1-1) void AppIO_out_msg_startMagicSquare();
1-2) void AppIO_out_msg_endMagicSquare();
1-3) int AppIO_in_order(void);
1-4) void AppIO_out_board(int anOrder,int aBoard[MAX_ORDER][MAX_ORDER]);

2) MagicSquare

2-1) void MagicSquare_setOrder(MagicSquare* _this, int anOrder);
2-2) Boolean MagicSquare_orderIsValid(MagicSquare* _this);
2-3) void MagicSquare_solve(MagicSquare* _this);
2-4) int MagicSquare_order(MagicSquare* _this);
2-5) int* MagicSquare_board(MagicSquare* _this);

2. 전체 코드

2-1. QuadEquation

1) main.c

```
//
//  main.c
//  CP2_WEEK4
//
//  Created by stu2017s10 on 2017. 3. 28..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include <stdio.h>
#include "AppIO.h"
#include "Common.h"
#include "QuadEquationProblem.h"
#include "Roots.h"

int main() {
    QuadEquationProblem qeProblem;  // 객체 qeProblem 생성 : 문제 풀이 과정에서 필요한 모든 정보를 소유한다.
    QuadEquation    equation;   // 객체 equation 생성
    Solution    solution;   // 객체 solution 생성

    AppIO_out_msg_startSolvlingQuadEquation();// 이차방정식 풀이 시작 메시지 출력
    printf("\n");
    while( AppIO_in_solvingIsRequested() ) {    // 사용자로부터 이차방정식 풀이 여부를 입력을 받는다.
        AppIO_in_quadEquation(&equation._c0,&equation._c1,&equation._c2);   // 이차방정식의 계수를 입력받는다.
        QuadEquationProblem_setEquation (&qeProblem, equation) ; // 객체와 부가정보를 제공
```

```
        AppIO_out_quadEquation (equation._c0, equation._c1, equation._c2) ; //
주어진 이차방정식을 보여준다.
        float determinant = QuadEquationProblem_determinant(&qeProblem);
        AppIO_out_determinant(determinant); //판별식의 값 출력

        if(QuadEquationProblem_secondOrderTermCoefficientIsZero (&qeProblem))
{ // 이차항의 계수가 0이면
            AppIO_out_msg_secondOrderTermCoefficientIsZero();   // 이차항의 계수가
0이어서 이차방정식이 아니라는 메시지 출력
        }
        else {
            if( QuadEquationProblem_determinantIsNegative(&qeProblem)) {    //
판별식의 값이 음수라면
                float determinant = QuadEquationProblem_determinant(&qeProblem);
                AppIO_out_msg_determinantIsNegative(determinant);    // 판별식의
값과 판별식의 값이 음수라는 메시지 출력
            }
            else {
                QuadEquationProblem_solve(&qeProblem);  // 방정식의 해를 구한다
                solution = QuadEquationProblem_solution(&qeProblem);
                AppIO_out_roots(solution._root1,solution._root2);   // 두 개의 방
정식 해를 출력
            }
        }
    }
          AppIO_out_msg_endSolvingQuadEquation();   //종료 메시지 출력
          return 0;
}
```

2) AppIO.c

```
//
//  AppIO.c
//  CP2_WEEK4
//
//  Created by stu2017s10 on 2017. 3. 28..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

//메시지 출력 함수
void AppIO_out_msg_startSolvlingQuadEquation() {    // <<<이차방정식 풀이>>> 메시지 출
력
    printf("<<<이차방정식 풀이>>>");
}
void AppIO_out_msg_endSolvingQuadEquation(){    // <<<이차방정식 풀이를 종료합니다>>>
메시지 출력
    printf("<<<이차방정식 풀이를 종료합니다>>>");
}

void AppIO_out_msg_secondOrderTermCoefficientIsZero (void) {    // 이차항의 계수가
0임을 알리는 메시지 출력
    printf(">이차항의 계수가 0이어서, 이차방정식이 아닙니다.\n");
}
//판별식의 값을 보여준다
void AppIO_out_msg_determinantIsNegative (float aDeterminant) { // 판별식의 값이 음
수임을 알리는 메시지 출력
```

```
    printf(">판별식의 값이 음수여서 해가 존재하지 않습니다.\n");
}

//입력 받는 함수
Boolean AppIO_in_solvingIsRequested (void){ // 이차방정식 풀이를 할 것인지 종료할 것인지 묻는 메시지 출력
    char answer;
    printf("방정식을 풀려면 y, 풀이를 종료하려면 아무 키나 입력하시오 : ");
    fpurge(stdin);  // 이전에 입력되어 임시 보관중인 모든 입력 값을 없앤다.
    answer = getchar(); // 키보드로부터 키 한개를 입력받는다.
    return ( answer == 'y');    // 입력 받은 것이 y이면 TRUE,그 외의 값은 FALSE 반환
}

//판별식의 값을 보여준다.
void AppIO_out_determinant(float aDeterminant) {
    printf(">판별식의 값 : %0.1f\n",aDeterminant);

}
//계수를 입력 받아서 이차방정식 객체를 만든다
void AppIO_in_quadEquation (float* p_c0, float* p_c1, float* p_c2){ // 계수를 각각 입력받아 돌려준다.
    printf("2차항의 계수를 입력하시오 : ");
    scanf("%f",p_c2);   // 2차항의 계수

    printf("1차항의 계수를 입력하시오 : ");
    scanf("%f",p_c1);   // 1차항의 계수

    printf("상수항의 계수를 입력하시오 : ");
    scanf("%f",p_c0);   // 상수항의 계수
}



//실행 결과를 보여주는 함수
// 계수의 값이 0인 항은 보이지 않는다 ( 모든 계수가 0이면 "0=0" 을 출력 )
//주어진 방정식을 출력한다.
void AppIO_out_quadEquation (float c0, float c1, float c2){
    Boolean nonZeroTermDoesExist = FALSE;
    printf(">주어진 방정식 : ");
    // 매크로를 사용하여 0인지 아닌지 검사한다
    if( !(FloatValueIsZero(c2)) ) { // 2차항의 계수가 0이 아니라면
        nonZeroTermDoesExist = TRUE;
        printf("(%f)x * x",c2);
    }
    if( !(FloatValueIsZero(c1)) ) { // 1차항의 계수가 0이 아니라면
        if(nonZeroTermDoesExist)
            printf(" + ");
        nonZeroTermDoesExist = TRUE;
        printf("(%f)x",c1);
    }
    if ( ! (FloatValueIsZero(c0)) ) {   // 상수항의 계수가 0이 아니라면
        if(nonZeroTermDoesExist)
            printf(" + ");
        nonZeroTermDoesExist = TRUE;
        printf("(%f)",c0);
    }
    if( !nonZeroTermDoesExist)  // 모든 계수가 0이면
        printf("0");
    printf(" = 0\n");

}
//방정식의 해를 보여준다
```

```
void AppIO_out_roots (float root1, float root2){
    printf(">방정식의 해는 다음과 같습니다.\n");
    printf("x1 = %0.2f\n",root1);
    printf("x2 = %0.2f\n",root2);
}
```

3) AppIO.h

```
//
//  AppIO.h
//  CP2_WEEK4
//
//  Created by stu2017s10 on 2017. 3. 28..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppIO_h
#define AppIO_h

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

//메시지 출력 함수
void AppIO_out_msg_startSolvlingQuadEquation();    // <<<이차방정식 풀이>>> 메시지 출력
void AppIO_out_msg_endSolvingQuadEquation();    // <<<이차방정식 풀이를 종료합니다>>> 메시지 출력
void AppIO_out_msg_secondOrderTermCoefficientIsZero(void) ;    // 이차항의 계수가 0임을 알리는 메시지 출력
void AppIO_out_msg_determinantIsNegative (float aDeterminant) ; // 판별식의 값이 음수임을 알리는 메시지

//입력 받는 함수
Boolean AppIO_in_solvingIsRequested (void) ; // 이차방정식 풀이를 할 것인지 종료할 것인지 묻는 메시지 출력
void AppIO_in_quadEquation (float* p_c0, float* p_c1, float* p_c2) ; // 계수를 각각 입력받아 돌려준다.

//실행 결과를 보여주는 함수
void AppIO_out_quadEquation (float c0, float c1, float c2) ;      // 주어진 방정식을 출력한다.
void AppIO_out_determinant(float aDeterminant);//판별식의 값을 보여준다.
void AppIO_out_roots (float root1, float root2) ;//방정식의 해를 보여준다

#endif /* AppIO_h */
```

3) QuadEquationProblem.c

```
//
//  QuadEquationProblem.c
//  CP2_WEEK4
//
//  Created by stu2017s10 on 2017. 3. 28..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

//객체에게 정보를 제공하는 함수
void QuadEquationProblem_setEquation (QuadEquationProblem* _this, QuadEquation
anEquation){
    _this->_equation = anEquation;
}

//객체의 상태를 검사하는 함수
Boolean
QuadEquationProblem_secondOrderTermCoefficientIsZero(QuadEquationProblem* _this)
{
    return FloatValueIsZero(_this->_equation._c2);
}

Boolean QuadEquationProblem_determinantIsNegative (QuadEquationProblem*_this)
{ // 판별식이 0인지 아닌지 상태를 보여주는 함수
    if(((_this->_equation._c1 *_this->_equation._c1)-4 *_this->_equation._c2
*_this->_equation._c0)<0) {
        return TRUE;
    }
    else return FALSE;
}

//객체의 내용을 변경시키는 함수

void QuadEquationProblem_solve (QuadEquationProblem* _this){   // 방정식의 해를 구한
다.
    float sqrtDeterminant;

    sqrtDeterminant = sqrt(QuadEquationProblem_determinant(_this));
    //결과를 _this->solution에 저장
    _this->_solution._root1 = (-_this->_equation._c1 + sqrtDeterminant )/
(2.0*_this->_equation._c2);
    _this->_solution._root2 = (-_this->_equation._c1 - sqrtDeterminant )/
(2.0*_this->_equation._c2);
}

//객체의 내용을 얻어내는 함수
QuadEquation QuadEquationProblem_equation(QuadEquationProblem* _this){
    return _this->_equation;
}

//주어진 이차방정식 문제 _this에 대해 판별식 계산
float QuadEquationProblem_determinant(QuadEquationProblem* _this){
    float determinant = (_this->_equation._c1 * _this->_equation._c1)-4 * _this-
>_equation._c2 * _this->_equation._c0;
    return determinant;
}

Solution QuadEquationProblem_solution(QuadEquationProblem* _this){  // 소유하고 있
는 정보로부터 객체의 상태를 얻는다.
```

```
    return _this->_solution;
}
```

4) QuadEquationProblem.h

```
//
//  QuadEquationProblem.h
//  CP2_WEEK4
//
//  Created by stu2017s10 on 2017. 3. 28..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef QuadEquationProblem_h
#define QuadEquationProblem_h

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

typedef struct{
    float _c0;  //상수항의 계수
    float _c1;  //1차항의 계수
    float _c2;  //2차항의 계수
}QuadEquation;  // 이차방정식 객체를 위한 자료형

typedef struct {
    QuadEquation    _equation;
    Solution        _solution;
}QuadEquationProblem;   // 이차방정식 문제 객체를 위한 자료형

//객체에게 정보를 제공하는 함수
void QuadEquationProblem_setEquation (QuadEquationProblem* _this, QuadEquation
anEquation) ;

//객체의 상태를 검사하는 함수(객체의 속성 정보 값이 바뀌지 않음)
Boolean
QuadEquationProblem_secondOrderTermCoefficientIsZero(QuadEquationProblem*
_this);   // 이차항이 0인지 아닌지 상태를 검사하는 함수
Boolean QuadEquationProblem_determinantIsNegative
(QuadEquationProblem*_this);  // 판별식이 0인지 아닌지 상태를 보여주는 함수

//객체의 내용을 변경시키는 함수
void QuadEquationProblem_solve (QuadEquationProblem* _this) ;   // 방정식의 해를 구
한다.

//객체의 내용을 얻어내는 함수
QuadEquation QuadEquationProblem_equation(QuadEquationProblem* _this);
float QuadEquationProblem_determinant(QuadEquationProblem* _this);  //주어진 이차방
정식 문제 _this에 대해 판별식 계산
Solution QuadEquationProblem_solution(QuadEquationProblem* _this); // 소유하고 있는
정보로부터 객체의 상태를 얻는다.
#endif /* QuadEquationProblem_h */
```

5) Roots.c

```
//
//  Roots.c
//  CP2_WEEK4
//
//  Created by stu2017s10 on 2017. 3. 28..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Roots.h"
```

6)Roots.h

```
//
//  Roots.h
//  CP2_WEEK4
//
//  Created by stu2017s10 on 2017. 3. 28..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Roots_h
#define Roots_h

typedef struct{
    float _root1;   // 방정식의 해 1
    float _root2;   // 방정식의 해 2
}Solution;  // 방정식의 해를 위한 자료형

#endif /* Roots_h */
```

7) Common.h

```
//
//  Common.h
//  CP2_WEEK4
//
//  Created by stu2017s10 on 2017. 3. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Common_h
#define Common_h


#include <math.h>

#define EPSILON 0.000001  // 상수에 이름을 부여하기 위한 매크로 선언
#define FloatValueIsZero(NUMBER)    fabsf(NUMBER)<EPSILON   // 반복 사용되는 코드를
위한 매크로 선언

typedef enum {FALSE,TRUE} Boolean;  // Boolean 타입을 define 해주고  FALSE, TRUE 값
을 가짐. FALSE와 TRUE의 순서는 바뀌면 안됨.



#endif /* Common_h */
```

2-2. MagicSquare

1) main.c

```
//
//  main.c
//  CP2_WEEK4_2
//
//  Created by stu2017s10 on 2017. 3. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//
#include <stdio.h>

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

int main(void) {

    MagicSquare magicSquare;    // 저장 장소 선언
    int order;
    magicSquare._maxOrder  = MAX_ORDER; // MAX_ORDER = 99

    AppIO_out_msg_startMagicSquare();   // 마방진 풀이 시작 메시지
    order = AppIO_in_order();  // 마방진 차수를 입력 받아 _order에 저장

    while(order != END_OF_RUN) {   // 마방진 차수가 -1이면 프로그램 종료, -1이 아니면 풀이 시작
        MagicSquare_setOrder(&magicSquare,order);   // 객체의 속성값을 설정하는 함수
        if( MagicSquare_orderIsValid(&magicSquare)) { // 차수가 유효한지 검사
            MagicSquare_solve(&magicSquare);  // 주어진 차수의 마방진을 푼다.
            AppIO_out_board(MagicSquare_order(&magicSquare),(int(*)
[MAX_ORDER])MagicSquare_board(&magicSquare)); // 마방진 판을 화면에 보여준다.
                    }
        order = AppIO_in_order();  // 다음 마방진을 위해 차수를 입력받는다.
    }
    AppIO_out_msg_endMagicSquare(); // 마방진 풀이 종료 메시지
    return 0;
}
```

2) AppIO.c

```
//
//  AppIO.c
//  CP2_WEEK4_2
//
//  Created by stu2017s10 on 2017. 3. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include <stdio.h>

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


void AppIO_out_msg_startMagicSquare() { // <<<마방진 풀이를 시작합니다>>> 메시지 출력
    printf("<<<마방진 풀이를 시작합니다>>>\n");
}

void AppIO_out_msg_endMagicSquare() {   // <<<마방진 풀이를 종료합니다>>> 메시지 출력
    printf("<<<마방진 풀이를 종료합니다>>>\n");
}
int AppIO_in_order(void) {  // 차수를 입력받기 위한 메시지를 내보내고 차수를 입력받아 얻는다.
```

```
    int _order;
    printf("마방진 차수를 입력하시오 : ");
    scanf("%d" ,&_order);   // _order = 차수
    return _order;
}

void AppIO_out_board(int anOrder,int aBoard[MAX_ORDER][MAX_ORDER]) { // 주어진 차
수의 완성된 마방진을 화면에 보여준다.
    printf("Magic Square Board : Order %d" ,anOrder);
    printf("\n");
    printf("%5s","");
    for( int col=0; col<anOrder; col++ ) {  // 차수만큼 col 증가 및 출력
        printf("[%2d]",col);
    }
    printf("\n");

    for( int row = 0; row<anOrder; row++ ){ // 차수만큼 row증가 및 출력
        printf("[%2d]",row);
        for( int col=0; col<anOrder; col++ ) {
            printf("%4d" ,aBoard[row][col]);    // aBoard에 저장된 row, col 값 출력
-> 마방진
        }
        printf("\n");
    }
}
```

3) AppIO.h

```
//
//  AppIO.h
//  CP2_WEEK4_2
//
//  Created by stu2017s10 on 2017. 3. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppIO_h
#define AppIO_h

#include "Common.h"

void AppIO_out_msg_startMagicSquare(); // <<<마방진 풀이를 시작합니다>>> 메시지 출력
void AppIO_out_msg_endMagicSquare();   // <<<마방진 풀이를 종료합니다>>> 메시지 출력
int AppIO_in_order(void);  // 차수를 입력받기 위한 메시지를 내보내고 차수를 입력받아 얻는다.
void AppIO_out_board(int anOrder,int aBoard[MAX_ORDER][MAX_ORDER]); // 주어진 차수
의 완성된 마방진을 화면에 보여준다.


#endif /* AppIO_h */
```

4) MagicSquare.c

```
//
//  MagicSquare.c
//  CP2_WEEK4_2
//
//  Created by stu2017s10 on 2017. 3. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

#define EMPTY_CELL -1

void MagicSquare_setOrder(MagicSquare* _this, int anOrder) {      // 객체의 속성값을
설정하는 함수
    _this->_order = anOrder;
}
//객체의 상태를 검사하는 함수
Boolean MagicSquare_orderIsValid(MagicSquare* _this) { // 주어진 차수가 유효한지 검사하
고, 유효하지 않다면 오류 메시지를 출력
    if( _this->_order < 3) {   // 차수가 3보다 작을때
        printf("오류 : 차수가 너무 작습니다. 3보다 크거나 같아야 합니다.\n");
        return FALSE;
    }
    else if( _this->_order > _this->_maxOrder) {    // 차수가 99보다 클 때
        printf("오류 : 차수가 너무 큽니다. %d보다 작아야 합니다.\n",_this->_maxOrder);
        return FALSE;
    }
    else if(( _this->_order %2)==0) {  // 차수가 짝수일 때
        printf("오류 : 차수가 짝수입니다. 차수는 홀수이어야 합니다\n");
        return FALSE;
    }
    else {
        return TRUE;
    }
}

void MagicSquare_solve(MagicSquare* _this) {   // 주어진 차수의 마방진을 푸는 함수
    int row,col;    // 위치 표현을 위한 변수

    CellLocation currentLoc;    // CellLocation의 currentLoc 객체 선언
    CellLocation nextLoc;    // CellLocation의 nextLoc 객체 선언


    // 보드 초기화
    for( row=0; row< _this->_order; row++ ) {
        for( col=0; col<_this->_order; col++ ) {
            _this->_board[row][col] = EMPTY_CELL;
        }
    }
    // 보드 채우기

    currentLoc._row = 0;    // 맨 윗줄
    currentLoc._col = _this->_order/2;    // 한 가운데 열

    int cellValue = 1;
    _this->_board[currentLoc._row][currentLoc._col] = cellValue;   // 보드의 현재
위치에  cellValue를 넣는다.
    int lastCellValue = _this->_order * _this->_order;
    cellValue = 2;
```

```
    for( cellValue = 2; cellValue <= lastCellValue; cellValue++ ) { // cellValue
가 2부터 (aMagicSquare._order * aMagicSquare._order)까지 증가하며 내용 반복

        // 현재 위치로부터 다음 위치인 오른쪽 위 위치를 계산한다.
        nextLoc._row = currentLoc._row -1;  // 다음 row = 현재 row - 1
        if( nextLoc._row <0)
            nextLoc._row = _this->_order-1;   // 맨 밑줄 위치로

        nextLoc._col = currentLoc._col +1;  // 다음 col = 현재 col + 1
        if( nextLoc._col >= _this->_order )
            nextLoc._col = 0;   // 가장 왼쪽 열 위치로


        nextLoc._col = (currentLoc._col+1) % _this->_order ;

        // 다음 위치가 채워져 있으면 바로 아래칸을 다음 위치로 수정한다.
        if( _this->_board[nextLoc._row][nextLoc._col] != EMPTY_CELL ) {
            nextLoc._row = currentLoc._row+1;
            nextLoc._col = currentLoc._col;
        }

        currentLoc = nextLoc;   // 다음 위치를 새로운 현재 위치로 한다.
        _this->_board[currentLoc._row][currentLoc._col] = cellValue;   // 보드의
새로운 현재위치에 cellValue를 넣는다.
    }
}
//객체의 정보를 얻는 함수
int MagicSquare_order( MagicSquare* _this ) {
    return _this->_order;
}

int* MagicSquare_board( MagicSquare* _this ) {
    return (int*) (_this->_board);  // 2차원 배열을 1차원 배열로 생각하도록 (int*)로 자료
형을 맞춘다.
}
```

5) MagicSquare.h

```
//
//  MagicSquare.h
//  CP2_WEEK4_2
//
//  Created by stu2017s10 on 2017. 3. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef MagicSquare_h
#define MagicSquare_h

#include "Common.h"

typedef struct {    // MagicSqure 객체 생성
    int _order; // 차수
    int _maxOrder;
    int _board[MAX_ORDER][MAX_ORDER];   // 보드
} MagicSquare;

typedef struct {    // CellLocation 객체 생성
    int _row;
    int _col;
} CellLocation;
```

```
void MagicSquare_setOrder(MagicSquare* _this, int anOrder); // 객체의 속성값을 설정하는 함수

//객체의 상태를 검사하는 함수
Boolean MagicSquare_orderIsValid(MagicSquare* _this); // 주어진 차수가 유효한지 검사하고, 유효하지 않다면 오류 메시지를 출력한다.
void MagicSquare_solve(MagicSquare* _this);    // 주어진 차수에 따라 마방진 판을 채운다.

//객체의 정보를 얻는 함수
int MagicSquare_order( MagicSquare* _this );
int* MagicSquare_board( MagicSquare* _this );

#endif /* MagicSquare_h */
```

6) Common.h

```
//
//  Common.h
//  CP2_WEEK4_2
//
//  Created by stu2017s10 on 2017. 3. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Common_h
#define Common_h

#define END_OF_RUN  -1  // -1이 입력되면 프로그램 종료
#define MAX_ORDER   99  // 차수는 최대 99로 정의

typedef enum {FALSE,TRUE} Boolean;  // FALSE와 TRUE 값을 갖는 Boolean 선언


#endif /* Common_h */
```

3. 종합 설명

3-1) QuadEquation

1. 객체 qeProblem, equation,solution을 생성한다.

2. AppIO_out_msg_startSolvlingQuadEquation() 함수로 이차방정식 풀이 시작 메시지를 출력한다.

3. while문을 통해 사용자로부터 이차방정식 풀이 여부를 입력받고, y가 입력되면 풀이를 시작하고 그 외의 값이 입력되면 풀이를 종료한다.

4. AppIO_in_quadEquation(&equation._c0,&equation._c1,&equation._c2)으로 이차방정식의 계수를 입력받는다.

5. QuadEquationProblem_setEquation (&qeProblem, equation)로 객체와 부가정보를 제공한다.

6. AppIO_out_quadEquation (equation._c0, equation._c1, equation._c2)으로 주어진 이차방정식을 출력한다.

7. float determinant의 값은 QuadEquationProblem_determinant(&qeProblem) 함수로 계산된 판별식의 값이다.

8. AppIO_out_determinant(determinant)로 판별식의 값을 출력한다.

9. 만약 사용자로부터 입력받은 이차항의 계수가 0이라면 AppIO_out_msg_secondOrderTermCoefficientIsZero() 함수가 이차항의 계수가 0이어서 이차방정식이 아니라는 메시지를 출력한다.

10. 이차항의 계수가 0이 아니지만 판별식의 값이 음수라면 AppIO_out_msg_determinantIsNegative(determinant) 함수로 판별식의 값이 음수라는 메시지를 출력한다.

11. 이차항의 계수가 0이 아니고, 판별식의 값도 음수가 아니라면 QuadEquationProblem_solve(&qeProblem)함수로 방정식의 해를 구한다.

12. AppIO_out_roots(solution._root1,solution._root2) 함수로 계산된 두 개의 방정식 해를 출력한다.

13. 다시 사용자로부터 이차방정식 풀이 여부를 묻고 y이 입력되면 다시 내용을 반복하고 그 외의 값이 입력되면 AppIO_out_msg_endSolvingQuadEquation()함수로 종료 메시지를 출력한다.

3-2) MagicSquare

1. 객체 magicSquare를 생성한다.

2. AppIO_out_msg_startMagicSquare() 함수로 마방진 풀이 시작 메시지를 출력한다.

3. AppIO_in_order() 함수로 마방진 차수를 입력 받아 _order에 저장한다.

4. END_OF_RUN은 -1이다. 입력된 마방진 차수가 -1이면 프로그램을 종료하고, -1이 아니면 풀이를 시작한다.

5. MagicSquare_setOrder(&magicSquare,order) 함수로 객체의 속성값을 설정해준다.

6. MagicSquare_orderIsValid(&magicSquare) 함수로 차수가 유효한지 검사하고 차수가 3보다 작을 경우 “오류 : 차수가 너무 작습니다. 3보다 크거나 같아야 합니다” 라는 오류 메시지를 출력하고 차수가 99보다 클 경우 “오류 : 차수가 너무 큽니다. MAX_ORDER 보다 작아야 합니다.”라는 오류 메시지를 출력, 차수가 짝수일 경우 “오류 : 차수가 짝수입니다. 차수는 홀수이어야 합니다.” 라는 오류 메시지를 출력한다.

7. 입력된 마방진 차수가 올바르다면 MagicSquare_solve(&magicSquare) 함수로 주어진 차수의 마방진을 푼다.

8. AppIO_out_board(MagicSquare_order(&magicSquare),(int(*)[MAX_ORDER])MagicSquare_board(&magicSquare)) 함수로 마방진 판을 화면에 보여준다.

9. AppIO_in_order()로 다시 다음 마방진을 위해 차수를 입력받는다.

10. -1이 입력되면 AppIO_out_msg_endMagicSquare() 함수가 마방진 풀이 종료 메시지를 출력한다.

4. 실행 결과

4-1. QuadEquation

```
<<<이차방정식 풀이>>>
방정식을 풀려면 y, 풀이를 종료하려면 아무 키나 입력하시오 : y
2차항의 계수를 입력하시오 : 1.0
1차항의 계수를 입력하시오 : 0.0
상수항의 계수를 입력하시오 : -1.0
>주어진 방정식 : (1.000000)x * x + (-1.000000) = 0
>판별식의 값 : 4.0
>방정식의 해는 다음과 같습니다.
x1 = 1.00
x2 = -1.00
방정식을 풀려면 y, 풀이를 종료하려면 아무 키나 입력하시오 : y
2차항의 계수를 입력하시오 : 1.0
1차항의 계수를 입력하시오 : 1.0
상수항의 계수를 입력하시오 : 1.0
>주어진 방정식 : (1.000000)x * x + (1.000000)x + (1.000000) = 0
>판별식의 값 : -3.0
>판별식의 값이 음수여서 해가 존재하지 않습니다.
방정식을 풀려면 y, 풀이를 종료하려면 아무 키나 입력하시오 : y
2차항의 계수를 입력하시오 : 0.0
1차항의 계수를 입력하시오 : 1.0
상수항의 계수를 입력하시오 : 1.0
>주어진 방정식 : (1.000000)x + (1.000000) = 0
>판별식의 값 : 1.0
>이차항의 계수가 0이어서, 이차방정식이 아닙니다.
방정식을 풀려면 y, 풀이를 종료하려면 아무 키나 입력하시오 : y
2차항의 계수를 입력하시오 : -4.0
1차항의 계수를 입력하시오 : 1.0
상수항의 계수를 입력하시오 : 0.0
>주어진 방정식 : (-4.000000)x * x + (1.000000)x = 0
>판별식의 값 : 1.0
>방정식의 해는 다음과 같습니다.
x1 = -0.00
x2 = 0.25
방정식을 풀려면 y, 풀이를 종료하려면 아무 키나 입력하시오 : n
<<<이차방정식 풀이를 종료합니다>>>Program ended with exit code: 0
```

4-2. MagicSquare

```
<<<마방진 풀이를 시작합니다>>>
마방진 차수를 입력하시오 : 2
오류 : 차수가 너무 작습니다. 3보다 크거나 같아야 합니다.
마방진 차수를 입력하시오 : 5
Magic Square Board : Order 5
      [ 0][ 1][ 2][ 3][ 4]
[ 0]  17  24   1   8  15
[ 1]  23   5   7  14  16
[ 2]   4   6  13  20  22
[ 3]  10  12  19  21   3
[ 4]  11  18  25   2   9
마방진 차수를 입력하시오 : 6
오류 : 차수가 짝수입니다. 차수는 홀수이어야 합니다
마방진 차수를 입력하시오 : 100
오류 : 차수가 너무 큽니다. 99보다 작아야 합니다.
마방진 차수를 입력하시오 : 3
Magic Square Board : Order 3
      [ 0][ 1][ 2]
[ 0]   8   1   6
[ 1]   3   5   7
[ 2]   4   9   2
마방진 차수를 입력하시오 : 7
Magic Square Board : Order 7
      [ 0][ 1][ 2][ 3][ 4][ 5][ 6]
[ 0]  30  39  48   1  10  19  28
[ 1]  38  47   7   9  18  27  29
[ 2]  46   6   8  17  26  35  37
[ 3]   5  14  16  25  34  36  45
[ 4]  13  15  24  33  42  44   4
[ 5]  21  23  32  41  43   3  12
[ 6]  22  31  40  49   2  11  20
마방진 차수를 입력하시오 : -1
<<<마방진 풀이를 종료합니다>>>
Program ended with exit code: 0
```