컴퓨터 프로그래밍 2
-hw05-

학번 : 201602038
이름 : 이 미 진

1. 함수 설명

1-1. 프로그램 05-1

1) AppIO

1-1) void AppIO_out_msg_startMagicSquare() : <<<마방진 풀이를 시작합니다>>> 메시지 출력

1-2) void AppIO_out_msg_endMagicSquare() : <<<마방진 풀이를 종료합니다>>> 메시지 출력

1-3) int AppIO_in_order(void) : 차수를 입력받기 위한 메시지를 내보내고 차수를 입력받아 얻는다.

1-4) void AppIO_out_board(int anOrder,int aBoard[MAX_ORDER][MAX_ORDER]) : 주어진 차수의 완성된 마방진을 화면에 보여준다.

2) MagicSquare

2-1) void MagicSquare_setOrder(MagicSquare* _this, int anOrder) : 객체의 속성값을 설정하는 함수

2-2) MagicSquare* MagicSquare_new() : 객체 생성

2-3) Boolean MagicSquare_orderIsValid(MagicSquare* _this) : 주어진 차수가 유효한지 검사하고, 유효하지 않다면 오류 메시지를 출력한다.

2-4) void MagicSquare_solve(MagicSquare* _this) : 주어진 차수에 따라 마방진 판을 채운다.

2-5) int MagicSquare_order(MagicSquare* _this) : order의 정보를 얻는다.

2-6) int* MagicSquare_board(MagicSquare* _this) : board의 정보를 얻는다.

2-7) void MagicSquare_delete(MagicSquare* _this) : 객체를 소멸시킨다.

1-2. 프로그램 05-2 (마방진 성능측정)

1) MagicSquare

1-1) void MagicSquare_setOrder(MagicSquare* _this, int anOrder) : 객체의 속성값을 설정하는 함수

1-2) MagicSquare* MagicSquare_new() : 객체 생성

1-3) Boolean MagicSquare_orderIsValid(MagicSquare* _this) : 주어진 차수가 유효한지 검사하고, 유효하지 않다면 오류 메시지를 출력한다.

1-4) void MagicSquare_solve(MagicSquare* _this) : 주어진 차수에 따라 마방진 판을 채운다.

1-5) int MagicSquare_order(MagicSquare* _this) : order의 정보를 얻는다.

1-6) int* MagicSquare_board(MagicSquare* _this) : board의 정보를 얻는다.

1-7) void MagicSquare_delete(MagicSquare* _this) : 객체를 소멸시킨다.

2) Timer

2-1) Timer* Timer_new(void) : 타이머 생성

2-2) void Timer_delete(Timer* _this) : 타이머 소멸

2-3) void Timer_start(Timer* _this) : 타이머 작동 시작

2-4) void Timer_stop(Timer* _this) : 타이머 작동 중지

2-5) long Timer_duration(Timer* _this) : 타이머 작동 시작부터 중지까지의 시간

3) AppIO

3-1) void AppIO_output(char* aMessage) : 개행이 없는 문자열을 출력

3-2) void AppIO_outputLine(char* aMessage) : 개행이 있는 문자열을 출력

3-3) void AppIO_outputExecutionTime(int anOrder, long anExecutionTime) : 성능 측정 값 출력

1-3. 프로그램 05-3

1) AppIO

1-1) void AppIO_out_msg_startMagicSquare() : <<<마방진 풀이를 시작합니다>>> 메시지 출력

1-2) void AppIO_out_msg_endMagicSquare() : <<<마방진 풀이를 종료합니다>>> 메시지 출력

1-3) int AppIO_in_order(void) : 차수를 입력받기 위한 메시지를 내보내고 차수를 입력받아 얻는다.

1-4) void AppIO_out_boardColumnTitle(int anOrder) : aCol 출력

1-5) void AppIO_out_boardRowTitle(int aRow) : aRow출력

1-6) void AppIO_out_cell(int aCell) : aCell에 저장된 마방진 출력

1-7) void AppIO_out_newLine(void) : 개행

2) MagicSqure

2-1) void MagicSquare_setOrder(MagicSquare* _this, int anOrder) : 객체의 속성값을 설정하는 함수

2-2) MagicSquare* MagicSquare_new() : 객체 생성

2-3) Boolean MagicSquare_orderIsValid(MagicSquare* _this) : 주어진 차수가 유효한지 검사하고, 유효하지 않다면 오류 메시지를 출력한다.

2-4) void MagicSquare_solve(MagicSquare* _this) : 주어진 차수에 따라 마방진 판을 채운다.

2-5) int MagicSquare_order(MagicSquare* _this) : order의 정보를 얻는다.

2-6) int* MagicSquare_board(MagicSquare* _this) : board의 정보를 얻는다.

2-7) void MagicSquare_delete(MagicSquare* _this) : 객체를 소멸시킨다.

3) Timer

3-1) Timer* Timer_new(void) : 타이머 생성

3-2) void Timer_delete(Timer* _this) : 타이머 소멸

3-3) void Timer_start(Timer* _this) : 타이머 작동 시작

3-4) void Timer_stop(Timer* _this) : 타이머 작동 중지

3-5) long Timer_duration(Timer* _this) : 타이머 작동 시작부터 중지까지의 시간

4) AppController

4-1) AppController* AppController_new() : 객체 생성

4-2) void AppController_delete(AppController* _this) : 더 이상 사용할 필요가 없는 AppController 객체 삭제

4-3) void AppController_showBoard(AppController* _this) : 마방진 판을 보여준다.

4-4) void AppController_run(AppController* _this) : app을 실행시킨다.

2. 전체 코드

2-1. 프로그램 05-1

1) main.c

```
//
//  main.c
//  CP2_WEEK5
//
//  Created by stu2017s10 on 2017. 4. 4..
//  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
  //  magicSquare->_maxOrder  = MAX_ORDER; // MAX_ORDER = 99

    AppIO_out_msg_startMagicSquare();    // 마방진 풀이 시작 메시지
    magicSquare = MagicSquare_new();     // magicSquare 객체 생성
    int order = AppIO_in_order();  // 마방진 차수를 입력 받아 _order에 저장

    while(order != END_OF_RUN) {    // 마방진 차수가 -1이면 프로그램 종료, -1이 아니면 풀이 시작
        // 첫번째 인수는 항상 객체의 사용권인 객체의 주소 값
        // 두번째 이후 인수는 별도의 정보가 필요할 경우에 제공되는 정보
        MagicSquare_setOrder(magicSquare,order);    // 객체의 속성값을 설정하는 함수
```

```
        if( MagicSquare_orderIsValid(magicSquare)) { // 차수가 유효한
지 검사
            MagicSquare_solve(magicSquare);  // 주어진 차수의 마방진을
푼다.
            AppIO_out_board(MagicSquare_order(magicSquare),(int(*)
[MAX_ORDER])MagicSquare_board(magicSquare)); // 마방진 판을 화면에 보여
준다.
        }
        order = AppIO_in_order();  // 다음 마방진을 위해 차수를 입력받는다.
    }
    MagicSquare_delete(magicSquare);    // 객체 소멸

    AppIO_out_msg_endMagicSquare(); // 마방진 풀이 종료 메시지
    return 0;
}
```

2) MagicSquare.c

```
//
//  MagicSquare.c
//  CP2_WEEK5_T
//
//  Created by stu2017s10 on 2017. 4. 5..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

#define EMPTY_CELL -1



//  객체 생성
MagicSquare* MagicSquare_new() {
    MagicSquare* _this;
    _this = NewObject(MagicSquare);
    return _this;
}

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 검사하고, 유효하지 않다면 오류 메시지를 출력
    _this->_maxOrder = MAX_ORDER;
    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 선
언
    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++ )
{ // cellValue가 2부터 (aMagicSquare._order * aMagicSquare._order)까
지 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 자료형을 맞춘다.
}

// 객체 소멸
void MagicSquare_delete(MagicSquare* _this) {
    free(_this);
}

int MagicSquare_cell(MagicSquare* _this, int aRow, int aCol) {
    if((0<=aRow && aRow < _this->_order) && (0<=aCol && aCol <
_this->_order)) {
```

```
        return _this->_board[aRow][aCol];
    }
    else {
        return  -1;
    }
}
```

3) MagicSquare.h

```
//
//  MagicSquare.h
//  CP2_WEEK5_T
//
//  Created by stu2017s10 on 2017. 4. 5..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//


#ifndef MagicSquare_h
#define MagicSquare_h

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

#define MAX_ORDER   99

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 생성
MagicSquare* MagicSquare_new();

//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 );


// 객체 소멸
void MagicSquare_delete(MagicSquare* _this);

int MagicSquare_cell(MagicSquare* _this, int aRow, int aCol);

#endif /* MagicSquare_h */
```

4) AppIO.c

```
//
//  AppIO.c
//  CP2_WEEK5
//
//  Created by stu2017s10 on 2017. 4. 4..
//  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 차수의 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 출
력
        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");
    }
}
```

5) AppIO.h

```
//
//  AppIO.h
//  CP2_WEEK5
//
//  Created by stu2017s10 on 2017. 4. 4..
//  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 입
력받아 얻는다.
void AppIO_out_board(int anOrder,int aBoard[MAX_ORDER]
[MAX_ORDER]); // 주어진 차수의 완성된 마방진을 화면에 보여준다.


#endif /* AppIO_h */
```

6) Common.h

```
//
//  Common.h
//  CP2_WEEK5
//
//  Created by stu2017s10 on 2017. 4. 4..
//  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
#define NewObject(TYPE) (TYPE*) malloc(sizeof(TYPE))

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


#endif /* Common_h */
```

2-2. 프로그램 05-2(마방진 성능측정)

1) main.c

```
//
//  main.c
//  CP2_WEEK5_T
//
//  Created by stu2017s10 on 2017. 4. 5..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

int main() {
    MagicSquare* magicSquare;   // 저장 장소 선언
    Timer* timer;
    int order,count;
```

```
    long executionTime; // 성능 측정 값

    timer = Timer_new();    // 타이머 생성

    AppIO_outputLine("<<<마방진 풀이의 성능 측정을 시작합니다>>>");
    AppIO_outputLine("하나의 차수에 대해 10회 반복 실행한 시간을 측정합니다");

    // MAX_ORDER = 99
    for( order = 9; order<=MAX_ORDER; order+=10) {  // 차수는 입력받지
않고 9부터 10씩 증가시키며 반복
        Timer_start(timer); // 타이머 작동 시작
        for( count=0; count<10; count++ ) { // count는 0부터 10까지 1
씩 증가
            magicSquare = MagicSquare_new();    // 객체 생성
            MagicSquare_setOrder(magicSquare,order);   // 객체의 속성
값을 설정하는 함수
            MagicSquare_solve(magicSquare);   // 주어진 차수의 마방진을
푸는 함수
            MagicSquare_delete(magicSquare);    // 객체 소멸
        }
        Timer_stop(timer);  // 타이머 작동 중지
        executionTime = Timer_duration(timer);
        AppIO_outputExecutionTime(order,executionTime); // 성능 측정
값 출력
    }
    Timer_delete(timer);    // 타이머 소멸
    AppIO_outputLine(">>>성능측정을 마칩니다<<<");
    return 0;
}
```

2) MagicSquare.c

```
//
//  MagicSquare.c
//  CP2_WEEK5_T
//
//  Created by stu2017s10 on 2017. 4. 5..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

#define EMPTY_CELL -1
```

```
//  객체 생성
MagicSquare* MagicSquare_new() {
    MagicSquare* _this;
    _this = NewObject(MagicSquare);
    return _this;
}

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 검사하고, 유효하지 않다면 오류 메시지를 출력
    _this->_maxOrder = MAX_ORDER;
    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 선
언
    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++ )
{ // cellValue가 2부터 (aMagicSquare._order * aMagicSquare._order)까
지 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 자료형을 맞춘다.
}
```

```
// 객체 소멸
void MagicSquare_delete(MagicSquare* _this) {
    free(_this);
}

int MagicSquare_cell(MagicSquare* _this, int aRow, int aCol) {
    if((0<=aRow && aRow < _this->_order) && (0<=aCol && aCol <
_this->_order)) {
        return _this->_board[aRow][aCol];
    }
    else {
        return  -1;
    }
}
```

3) MagicSquare.h

```
//
//  MagicSquare.h
//  CP2_WEEK5_T
//
//  Created by stu2017s10 on 2017. 4. 5..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//


#ifndef MagicSquare_h
#define MagicSquare_h

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

#define MAX_ORDER   99

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);   // 객
체의 속성값을 설정하는 함수
```

```
// 객체의 생성
MagicSquare* MagicSquare_new();

//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 );


// 객체 소멸
void MagicSquare_delete(MagicSquare* _this);

int MagicSquare_cell(MagicSquare* _this, int aRow, int aCol);

#endif /* MagicSquare_h */
```

4) Timer.c

```
//
//  Timer.c
//  CP2_WEEK5_T
//
//  Created by stu2017s10 on 2017. 4. 5..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Timer.h"
#include "Common.h"
#include <math.h>

Timer* Timer_new(void) { // 타이머 생성
    Timer* _this = NewObject(Timer);
    return _this;
}

void Timer_delete(Timer* _this) {    // 타이머 소멸
    free(_this);
}
void Timer_start(Timer* _this) { // 타이머 작동 시작
    _this->startCounter = clock();  // 실행 전 카운터 값을 얻음
}

void Timer_stop(Timer* _this) {  // 타이머 작동 중지
```

```
    _this->stopCounter = clock();   // 실행 후 카운터 값을 얻음
}

long Timer_duration(Timer* _this) {  // 타이머 작동 시작부터 중지까지의 시
간
    // CLOCKS_PER_SEC : 초 당 클럭 수를 나타내는 상수
    // *1.0E+6을 곱하여 마이크로 초로 변환
    return (long)(_this->stopCounter -_this-
>startCounter*(long)1000000 / (long)CLOCKS_PER_SEC);
}
```

5) Timer.h

```
//
//  Timer.h
//  CP2_WEEK5_T
//
//  Created by stu2017s10 on 2017. 4. 5..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Timer_h
#define Timer_h

#include <stdio.h>
#include <time.h>

typedef struct{
    clock_t startCounter;   // clock_t 는 cpu clock수를 저장하는 구조체
    clock_t stopCounter;
} Timer;

Timer* Timer_new(void); // 타이머 생성
void Timer_delete(Timer* _this);    // 타이머 소멸
void Timer_start(Timer* _this); // 타이머 작동 시작
void Timer_stop(Timer* _this);  // 타이머 작동 중지
long Timer_duration(Timer* _this);  // 타이머 작동 시작부터 중지까지의 시간

#endif /* Timer_h */
```

6) AppIO.c

```c
//
//  AppIO.c
//  CP2_WEEK5_T
//
//  Created by stu2017s10 on 2017. 4. 5..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "AppIO.h"

void AppIO_output(char* aMessage) {  // 배열을 받음 , == char
aMessage[]
    printf("%s", aMessage); // main에서 문자열을 받고 aMessage에 저장
}
void AppIO_outputLine(char* aMessage) { // \n 포함
    printf("%s\n", aMessage); // main에서 문자열을 받고 aMessage에 저장
}
void AppIO_outputExecutionTime(int anOrder, long anExecutionTime)
{ // 성능 측정 값 출력
    printf("차수 : %2d, 시간: %ld(마이크로 초)\n",
anOrder,anExecutionTime);
}
```

7) AppIO.h

```c
//
//  AppIO.h
//  CP2_WEEK5_T
//
//  Created by stu2017s10 on 2017. 4. 5..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppIO_h
#define AppIO_h

#include <stdio.h>

void AppIO_output(char* aMessage);  // 배열을 받음 , == char
aMessage[]
void AppIO_outputLine(char* aMessage);  // \n 포함
void AppIO_outputExecutionTime(int anOrder, long anExecutionTime);
// 성능 측정 값 출력

#endif /* AppIO_h */
```

8) Common.h

```
//
//  Common.h
//  CP2_WEEK5_T
//
//  Created by stu2017s10 on 2017. 4. 5..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Common_h
#define Common_h

#include <stdlib.h>

#define END_OF_RUN  -1  // -1이 입력되면 프로그램 종료
#define MAX_ORDER   99  // 차수는 최대 99로 정의
#define NewObject(TYPE) (TYPE*) malloc(sizeof(TYPE))

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


#endif /* Common_h */
```

2-2. 프로그램 05-3

1) main.c

```
//
//  main.c
//  CP2_WEEK5_2
//
//  Created by stu2017s10 on 2017. 4. 4..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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


int main() {
    AppController* appController;   // 저장 장소 선언
    appController = AppController_new();    // 객체 생성
    AppController_run(appController);    // app을 실행시킨다.
    AppController_delete(appController);   // 더 이상 사용할 필요가 없는
AppController 객체 삭제
    return 0;
}
```

2) AppIO.c

```
//
//  AppIO.c
//  CP2_WEEK5_2
//
//  Created by stu2017s10 on 2017. 4. 4..
//  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");
```

```c
}

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_boardColumnTitle(int anOrder) {
    printf("Magic Square Board : Order %d" ,anOrder);
    printf("\n");
    printf("%5s","");
    for( int aCol=0; aCol<anOrder; aCol++) { //차수만큼 aCol 증가 및 출
력
        printf("[%2d]",aCol);
    }
    printf("\n");
}
void AppIO_out_boardRowTitle(int aRow) {    // aRow 출력
        printf("[%2d]",aRow);
}
void AppIO_out_cell(int aCell) {    // aCell에 저장된 마방진 출력
    printf("%4d" ,aCell);
}

void AppIO_out_newLine(void) {
    printf("\n");
}
```

3) AppIO.h

```
//
//  AppIO.h
//  CP2_WEEK5_2
//
//  Created by stu2017s10 on 2017. 4. 4..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppIO_h
#define AppIO_h

#include <stdio.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 입
력받아 얻는다.
void AppIO_out_board(int anOrder,int aBoard[MAX_ORDER]
[MAX_ORDER]); // 주어진 차수의 완성된 마방진을 화면에 보여준다.

void AppIO_out_boardColumnTitle(int anOrder);   // aCol 출력
void AppIO_out_boardRowTitle(int aRow); // aRow 출력
void AppIO_out_cell(int aCell);    // aCell에 저장된 마방진 출력
void AppIO_out_newLine(void);   // 개행



#endif /* AppIO_h */
```

4) MagicSquare.c

```
//
//  MagicSquare.c
//  CP2_WEEK5_2
//
//  Created by stu2017s10 on 2017. 4. 4..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

```
#define EMPTY_CELL -1



//  객체 생성
MagicSquare* MagicSquare_new() {
    MagicSquare* _this;
    _this = NewObject(MagicSquare);
    return _this;
}

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 검사하고, 유효하지 않다면 오류 메시지를 출력
    _this->_maxOrder = MAX_ORDER;
    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 선
언
    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++ )
{ // cellValue가 2부터 (aMagicSquare._order * aMagicSquare._order)까
지 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 자료형을 맞춘다.
}

// 객체 소멸
void MagicSquare_delete(MagicSquare* _this) {
    free(_this);
}

// AppController_showBoard에서 사용
int MagicSquare_cell(MagicSquare* _this, int aRow, int aCol) {
    if((0<=aRow && aRow < _this->_order) && (0<=aCol && aCol <
_this->_order)) {    // 위치가 정상일 때
        return _this->_board[aRow][aCol];
    }
    else {  // 위치가 정상 범위를 벗어났을 때
        return  -1;
    }
}
```

6) MagicSquare.h

```
//
//  MagicSquare.h
//  CP2_WEEK5_2
//
//  Created by stu2017s10 on 2017. 4. 4..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef MagicSquare_h
#define MagicSquare_h

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

#define MAX_ORDER   99

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 생성
MagicSquare* MagicSquare_new();

//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 );

// 객체 소멸
void MagicSquare_delete(MagicSquare* _this);

int MagicSquare_cell(MagicSquare* _this, int aRow, int aCol);   // 마방진 값

#endif /* MagicSquare_h */
```

7) Timer.c

```
//
//  Timer.c
//  CP2_WEEK5_2
//
//  Created by stu2017s10 on 2017. 4. 4..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Timer.h"
#include "Common.h"
#include <math.h>

Timer* Timer_new(void) {    // 타이머 생성
    Timer* _this = NewObject(Timer);
    return _this;
}

void Timer_delete(Timer* _this) {   // 타이머 소멸
    free(_this);
}
void Timer_start(Timer* _this) {    // 타이머 작동 시작
    _this->startCounter = clock();  // 실행 전 카운터 값을 얻음
```

```
}

void Timer_stop(Timer* _this) { // 타이머 작동 중지
    _this->stopCounter = clock();
}

long Timer_duration(Timer* _this) {
    return (long)(_this->stopCounter -_this-
>stopCounter*(long)1000000 / (long)CLOCKS_PER_SEC);
}
```

8) Timer.h

```
//
//  Timer.h
//  CP2_WEEK5_2
//
//  Created by stu2017s10 on 2017. 4. 4..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Timer_h
#define Timer_h

#include <stdio.h>
#include <time.h>

typedef struct{
    clock_t startCounter;   // clock_t 는 cpu clock수를 저장하는 구조체
    clock_t stopCounter;
} Timer;

Timer* Timer_new(void); // 타이머 생성
void Timer_delete(Timer* _this);    // 타이머 소멸
void Timer_start(Timer* _this); // 타이머 작동 시작
void Timer_stop(Timer* _this);  // 타이머 작동 중지
long Timer_duration(Timer* _this);  // 타이머 작동 시작부터 중지까지의 시간

#endif /* Timer_h */
```

9) AppController.c

```
//
//  AppController.c
//  CP2_WEEK5_2
//
//  Created by stu2017s10 on 2017. 4. 4..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

AppController* AppController_new() {    // AppController 객체 생성
    AppController* _this = NewObject(AppController);
    _this->_magicSquare = MagicSquare_new();
    return _this;
}

void AppController_delete(AppController* _this) {   // 더 이상 사용할
필요가 없는 AppController 객체 삭제
    MagicSquare_delete(_this->_magicSquare);
    free(_this);
}

void AppController_showBoard(AppController* _this) {    // 마방진 판
을 보여준다
    int order = MagicSquare_order(_this->_magicSquare);
    AppIO_out_boardColumnTitle(order);

    for( int row = 0; row<order; row++ ) {
        AppIO_out_boardRowTitle(row);
        for(int col=0; col<order; col++ ) {
            AppIO_out_cell(MagicSquare_cell(_this-
>_magicSquare,row,col));
        }
        AppIO_out_newLine();    //개행
    }
}

void AppController_run(AppController* _this) {  // app을 실행시킨다.
    AppIO_out_msg_startMagicSquare();
    int order = AppIO_in_order();
    while ( order != END_OF_RUN ) {
        MagicSquare_setOrder(_this->_magicSquare, order);

        if (MagicSquare_orderIsValid(_this->_magicSquare)) {
            MagicSquare_solve(_this->_magicSquare);
            AppController_showBoard(_this); // 마방진 판을 보여준다.
```

```
        }
        order = AppIO_in_order();
    }
    AppIO_out_msg_endMagicSquare();
}
```

10) AppController.h

```
//
//  AppController.h
//  CP2_WEEK5_2
//
//  Created by stu2017s10 on 2017. 4. 4..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppController_h
#define AppController_h

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

typedef struct {
    MagicSquare* _magicSquare;
} AppController;    // AppController 객체 생성

AppController* AppController_new(); // 객체 생성
void AppController_delete(AppController* _this);   // 더 이상 사용할 필요가 없는 AppController 객체 삭제
void AppController_showBoard(AppController* _this); // 마방진 판을 보여준다.
void AppController_run(AppController* _this); // app을 실행시킨다.

#endif /* AppController_h */
```

11) Common.h

```
//
//  Common.h
//  CP2_WEEK5_2
//
//  Created by stu2017s10 on 2017. 4. 4..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Common_h
#define Common_h
```

```c
#include <stdlib.h>

#define END_OF_RUN  -1  // -1이 입력되면 프로그램 종료
#define MAX_ORDER   99  // 차수는 최대 99로 정의
#define NewObject(TYPE) (TYPE*) malloc(sizeof(TYPE))

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

#endif /* Common_h */
```

3. 전체 설명

3-1) 프로그램 05-1

1. magicSquare 저장 장소를 선언한다.
2. AppIO_out_msg_startMagicSquare()로 마방진 풀이 시작 메시지를 출력한다.
3. MagicSquare_new() 함수로 magicSquare 객체를 생성한다.
4. AppIO_in_order() 함수로 마방진 차수를 입력받는다.
5. 입력된 차수가 -1(END_OF_RUN)이 아니면 MagicSquare_setOrder(magicSquare,order)으로 객체의 속성값을 설정해주고 MagicSquare_orderIsValid(magicSquare)로 입력된 차수가 유효한지 검사한다.
6. 입력된 차수가 유효하지 않으면 오류 메시지를 출력하고, 유효하다면 MagicSquare_solve(magicSquare) 함수로 주어진 차수의 마방진을 푼다.
7. AppIO_out_board(MagicSquare_order(magicSquare),(int(*)[MAX_ORDER])MagicSquare_board(magicSquare)) 함수로 마방진 판을 화면에 보여준다.
8. AppIO_in_order()로 다음 마방진을 위해 차수를 입력받는다.
9. -1이 입력되어 while문의 반복이 끝나면 MagicSquare_delete(magicSquare) 함수가 객체를 소멸시키고 AppIO_out_msg_endMagicSquare()로 마방진 풀이 종료 메시지를 출력한다.

3-2) 프로그램 05-2

1. magicSquare와 timer 저장 장소를 선언한다.

2. Timer_new() 함수로 타이머를 생성한다.

3. AppIO_outputLine()로 개행이 있는 문자열을 출력한다. (<<<마방진 풀이의 성능 측정을 시작합니다>>>, 하나의 차수에 대해 10회 반복 실행한 시간을 측정합니다)

4. for문을 통해 차수는 입력받지 않고 9부터 차수를 10씩 증가시키며 99(MAX_ORDER)까지 반복한다.

5. Timer_start(timer)함수로 타이머 작동을 시작한다.

6. 이중 for문으로 차수가 증가할때 count도 1씩 증가한다.

7. MagicSquare_new()함수로 객체를 생성한다.

8. MagicSquare_setOrder() 함수로 객체의 속성값을 설정해준다.

9. MagicSquare_solve() 함수로 주어진 차수의 마방진을 푼다.

10. MagicSquare_delete() 함수에서 free를 통해 객체를 소멸시킨다.

11. count가 증가하면서 반복된 두번째 for문이 종료되면 Timer_stop(timer)로 타이머 작동을 중지한다.

12. Timer_duration(timer) 함수로 타이머 작동 시작부터 중지까지의 시간을 계산하고 executionTime에 저장한다.

13. AppIO_outputExecutionTime(order,executionTime) 함수로 성능 측정값을 출력한다.

14. 모든 반복이 끝나고 Timer_delete(timer) 함수가 타이머를 소멸시킨다.

15. AppIO_outputLine()함수로 개행이 있는 “>>>성능측정을 마칩니다<<< “ 메시지를 출력한다.

3-3) 프로그램 05-3

1. appController 저장 장소를 선언한다.

2. AppController_new() 함수로 객체를 생성한다.

3. AppController_run(appController) 함수로 app을 실행시킨다.

4. AppController_run()에서는 AppIO_out_msg_startMagicSquare() 함수로 시작 메시지를 출력하고, AppIO_in_order()로 차수를 입력받는다.

5. 입력 받은 차수가 -1(END_OF_RUN)이 아니라면 MagicSquare_setOrder()로 객체의 속성값을 설정하고, MagicSquare_orderIsValid() 함수로 입력된 차수가 유효한지 확인한다. 만약 차수가 유효하지 않다면 오류 메시지를 출력하고 다시 차수를 입력받는다.

6. 차수가 유효하다면 MagicSquare_solve() 함수로 주어진 차수의 마방진을 푼다.

7. AppController_showBoard() 함수로 마방진 판을 보여준다. 그리고 다시 차수를 입력받는다.

8. 입력된 차수가 -1이라면 프로그램을 종료하고 AppController_run() 함수도 종료된다.

9. AppController_delete() 함수에서 free를 통해 더 이상 사용할 필요가 없는 AppController 객체를 삭제한다.

4. 실행결과

1) 프로그램 05-1

```
<<<마방진 풀이를 시작합니다>>>
마방진 차수를 입력하시오 : 2
오류 : 차수가 너무 작습니다. 3보다 크거나 같아야 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 : 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 : 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 : 10
오류 : 차수가 짝수입니다. 차수는 홀수이어야 합니다
마방진 차수를 입력하시오 : 100
오류 : 차수가 너무 큽니다. 99보다 작아야 합니다.
마방진 차수를 입력하시오 : -1
<<<마방진 풀이를 종료합니다>>>
Program ended with exit code: 0
```

2) 프로그램 05-2

```
<<<마방진 풀이의 성능 측정을 시작합니다>>>
하나의 차수에 대해 10회 반복 실행한 시간을 측정합니다
차수 :  9, 시간: 34(마이크로 초)
차수 : 19, 시간: 87(마이크로 초)
차수 : 29, 시간: 260(마이크로 초)
차수 : 39, 시간: 356(마이크로 초)
차수 : 49, 시간: 559(마이크로 초)
차수 : 59, 시간: 845(마이크로 초)
차수 : 69, 시간: 1138(마이크로 초)
차수 : 79, 시간: 1453(마이크로 초)
차수 : 89, 시간: 1705(마이크로 초)
차수 : 99, 시간: 1927(마이크로 초)
>>>성능측정을 마칩니다<<<
Program ended with exit code: 0
```

3) 프로그램 05-3

```
<<<마방진 풀이를 시작합니다>>>
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 : 2
오류 : 차수가 너무 작습니다. 3보다 크거나 같아야 합니다.
마방진 차수를 입력하시오 : 100
오류 : 차수가 너무 큽니다. 99보다 작아야 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 : 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 : 10
오류 : 차수가 짝수입니다. 차수는 홀수이어야 합니다
마방진 차수를 입력하시오 : -1
<<<마방진 풀이를 종료합니다>>>
Program ended with exit code: 0
```

5. 프로그램 05-2의 차수와 시간의 관계 꺾은선 그래프

9	34	16.2
19	87	72.2
29	260	168.2
39	356	304.2
49	559	480.2
59	845	696.2
69	1138	952.2
79	1453	1248.2
89	1705	1584.2
99	1927	1960.2