컴퓨터 프로그래밍 2
-hw03-

학번 : 201602038
이름 : 이 미 진

1. 함수 설명

1-1 AppIO

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

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

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

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

1-2 MagicSquare

1) Boolean MagicSquare_orderIsValid(MagicSquare aMagicSquare) : 주어진 차수가 유효한지 검사하고, 유효하지 않다면 오류 메시지를 출력한다.

2) void MagicSquare_solve(MagicSquare aMagicSquare, int aBoard[MAX_ORDER][MAX_ORDER]) : 주어진 차수에 따라 마방진 판을 채운다.

2. 전체 코드

2-1) 객체를 사용하지 않은 코드

1) main.c

```
//
//  main.c
//  week3
//
//  Created by stu2017s10 on 2017. 3. 21..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

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

int main(void) {
    int _order;
    int _board[MAX_ORDER][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
        if( MagicSquare_orderIsValid(_order)) { // 차수가 유효한지 검사
            MagicSquare_solve(_order,_board);   //주어진 차수의 마방진을
얻는다.
            AppIO_out_board(_order,_board); // 마방진 판을 화면에 보여준
다.
        }
        _order = AppIO_in_order();  // 다음 마방진을 위해 차수를 입력받는
다.
    }
    AppIO_out_msg_endMagicSquare(); // 마방진 풀이 종료 메시지
    return 0;
}
```

2) Common.h

```
//
//  Common.h
//  week3
//
//  Created by stu2017s10 on 2017. 3. 21..
//  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) AppIO.c

```
//  AppIO.c
//  week3
//
//  Created by stu2017s10 on 2017. 3. 21..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//
```

```c
#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 보여준다1
    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");
    }
}
```

4) AppIO.h

```
//
//  AppIO.h
//  week3
//
//  Created by stu2017s10 on 2017. 3. 21..
//  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 차수의 완성된 마방진을 화면에 보여준다.

#endif /* AppIO_h */
```

5) MagicSquare.c

```
//
//  MagicSquare.c
//  week3
//
//  Created by stu2017s10 on 2017. 3. 21..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//
#include <stdio.h>
#include "MagicSquare.h"
#include "Common.h"
#define EMPTY_CELL -1



Boolean MagicSquare_orderIsValid(int anOrder) { // 주어진 차수가 유효한지 검사하고, 유효하지 않다면 오류 메시지를 출력
    if(anOrder<3) {   // 차수가 3보다 작을때
        printf("오류 : 차수가 너무 작습니다. 3보다 크거나 같아야 합니다.\n");
```

```
        return FALSE;
    }
    else if(anOrder>MAX_ORDER) {    // 차수가 99보다 클 때
        printf("오류 : 차수가 너무 큽니다. %d보다 작아야 합니다.
\n",MAX_ORDER);
        return FALSE;
    }
    else if((anOrder %2)==0) {  // 차수가 짝수일 때
        printf("오류 : 차수가 짝수입니다. 차수는 홀수이어야 합니다\n");
        return FALSE;
    }
    else {
    return TRUE;
    }
}

void MagicSquare_solve(int anOrder, int aBoard[MAX_ORDER]
[MAX_ORDER] ) {   // 주어진 차수의 마방진을 푸는 함수
    int row,col;    // 위치 표현을 위한 변수

    CellLocation currentLoc;
    CellLocation nextLoc;


    // 보드 초기화
    for( row=0; row<anOrder; row++ ) {
        for( col=0; col<anOrder; col++ ) {
            aBoard[row][col] = EMPTY_CELL;
        }
    }
    // 보드 채우기

    currentLoc._row = 0;    // 맨 윗줄
    currentLoc._col = anOrder/2;    // 한 가운데 열

    int cellValue = 1;
    aBoard[currentLoc._row][currentLoc._col] = cellValue;   // 보드
의 현재 위치에  cellValue를 넣는다.
    int lastCellValue = anOrder * anOrder;
    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 = 현재row-1
        if( nextLoc._row <0)
            nextLoc._row = anOrder-1;   // 맨 밑줄 위치로
```

```
        nextLoc._col = currentLoc._col +1;  // 다음 col = 현재col+1
        if( nextLoc._col >= anOrder )
            nextLoc._col = 0;   // 가장 왼쪽 열 위치로


        nextLoc._col = (currentLoc._col+1) % anOrder ;

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

        currentLoc = nextLoc;   // 다음 위치를 새로운 현재 위치로 한다.
       aBoard[currentLoc._row][currentLoc._col] = cellValue;   //
보드의 새로운 현재위치에 cellValue를 넣는다.
    }
}
```

6) MagicSquare.h

```
//
//  MagicSquare.h
//  week3
//
//  Created by stu2017s10 on 2017. 3. 21..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef MagicSquare_h
#define MagicSquare_h

#include "Common.h"

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

Boolean MagicSquare_orderIsValid(int anOrder); // 주어진 차수가 유효한지
검사하고, 유효하지 않다면 오류 메시지를 출력한다.
void MagicSquare_solve(int anOrder, int aBoard[MAX_ORDER]
[MAX_ORDER] );    // 주어진 차수에 따라 마방진 판을 채운다.

#endif /* MagicSquare_h */
```

2-2 객체를 사용한 코드

1) main.c

```
//
//  main.c
//  week3
//
//  Created by stu2017s10 on 2017. 3. 21..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include <stdio.h>

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

int main(void) {

    MagicSquare magicSquare;
    magicSquare._maxOrder = MAX_ORDER;  // MAX_ORDER = 99

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

    while(magicSquare._order != END_OF_RUN) {   // 마방진 차수가 -1이면
프로그램 종료
        if( MagicSquare_orderIsValid(magicSquare)) { // 차수가 유효한
지 검사
            MagicSquare_solve(magicSquare,magicSquare._board);  //
주어진 차수의 마방진을 푼다.

AppIO_out_board(magicSquare._order,magicSquare._board); // 마방진 판
을 화면에 보여준다.
        }
        magicSquare._order = AppIO_in_order();  // 다음 마방진을 위해
차수를 입력받는다.
    }
    AppIO_out_msg_endMagicSquare(); // 마방진 풀이 종료 메시지
    return 0;
}
```

2) AppIO.c

```
//
//  AppIO.c
//  week3
//
//  Created by stu2017s10 on 2017. 3. 21..
//  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");
    }
}
```

3) AppIO.h

```
//
//  AppIO.h
//  week3
//
//  Created by stu2017s10 on 2017. 3. 21..
//  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 차수의 완성된 마방진을 화면에 보여준다.

#endif /* AppIO_h */
```

4) Common.h

```
//
//  Common.h
//  week3
//
//  Created by stu2017s10 on 2017. 3. 21..
//  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 */
```

5) MagicSquare.c

```
//  MagicSquare.c
//  week3
//
//  Created by stu2017s10 on 2017. 3. 21..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//
#include <stdio.h>
#include "MagicSquare.h"
#include "Common.h"
#define EMPTY_CELL -1



Boolean MagicSquare_orderIsValid(MagicSquare aMagicSquare) {
    if(aMagicSquare._order < 3) {   // 차수가 3보다 작을때
        printf("오류 : 차수가 너무 작습니다. 3보다 크거나 같아야 합니다.\n");
        return FALSE;
    }
    else if(aMagicSquare._order>MAX_ORDER) {    // 차수가 99보다 클 때
        printf("오류 : 차수가 너무 큽니다. %d보다 작아야 합니다.
\n",MAX_ORDER);
        return FALSE;
    }
    else if((aMagicSquare._order %2)==0) {  // 차수가 짝수일 때
        printf("오류 : 차수가 짝수입니다. 차수는 홀수이어야 합니다.\n”);
        return FALSE;
    }
    else {
    return TRUE;
    }
}

void MagicSquare_solve(MagicSquare aMagicSquare, int
aBoard[MAX_ORDER][MAX_ORDER] ) {   // 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<aMagicSquare._order; row++ ) {
        for( col=0; col<aMagicSquare._order; col++ ) {
            aBoard[row][col] = EMPTY_CELL;
        }
    }
    // 보드 채우기

    currentLoc._row = 0;    // 맨 윗줄
    currentLoc._col = aMagicSquare._order/2;    // 한 가운데 열


    int cellValue = 1;
    aBoard[currentLoc._row][currentLoc._col] = cellValue;   // 보드
의 현재 위치에  cellValue를 넣는다.
    int lastCellValue = aMagicSquare._order * aMagicSquare._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 = aMagicSquare._order-1;   // 맨 밑줄 위치로

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


        nextLoc._col = (currentLoc._col+1) % aMagicSquare._order ;

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

        currentLoc = nextLoc;   // 다음 위치를 새로운 현재 위치로 한다.
        aBoard[currentLoc._row][currentLoc._col] = cellValue;   //
보드의 새로운 현재위치에 cellValue를 넣는다.
    }
}
```

6) MagicSquare.h

```
//
//  MagicSquare.h
//  week3
//
//  Created by stu2017s10 on 2017. 3. 21..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef MagicSquare_h
#define MagicSquare_h

#include "Common.h"

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

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

Boolean MagicSquare_orderIsValid(MagicSquare aMagicSquare); // 주어진 차수가 유효한지 검사하고, 유효하지 않다면 오류 메시지를 출력한다.
void MagicSquare_solve(MagicSquare aMagicSquare, int aBoard[MAX_ORDER][MAX_ORDER] );    // 주어진 차수에 따라 마방진 판을 채운다.


#endif /* MagicSquare_h */
```

3. 종합 설명(객체를 사용한 마방진 풀이 코드를 기준으로 작성)

1. main.c에서 MagicSquae의 magicSquare 객체를 생성한다.
2. AppIO_out_msg_startMagicSquare() 함수로 마방진 풀이 시작 메시지를 출력한다.
3. AppIO_in_order()로 마방진 차수를 입력받은 후 magicSquare._order에 저장
4. END_OF_RUN 은 -1이고, 만약 입력된 마방진의 차수가 -1이라면 프로그램을 종료하고, 아닐경우 코드를 반복한다.
5. 입력받은 차수를 MagicSquare_orderIsValid(magicSquare)를 통해 차수가 유효한지 검사한다.
6. 입력된 차수가 3보다 작을 경우 "차수가 너무 작습니다. 3보다 크거나 같아야 합니다" 오류 메시지를 출력하고, 99보다 클 경우 "차수가 너무 큽니다. MAX_ORDER(99)보다 작아야 합니다." 오류 메시지를 출력한다. 그리고 차수가 짝수일 경우 "차수가 짝수입니다. 차수는 홀수이어야 합니다." 오류 메시지 출력
7.차수가 유효하다면 MagicSquare_solve(magicSquare,magicSquare._board)를 통해 주어진 차수의 마방진을 푼다.
8. MagicSquare_solve(magicSquare,magicSquare._board)에서는 위치 표현을 위한 변수 row,col을 선언하고, CellLocation의 객체 currentLoc,nextLoc를 선언한다.
for문을 이용해 row와 col을 aMagicSquare._order만큼 반복하면서
EMPTY_CELL(-1)로 aBoard를 초기화 한다.
처음 보드를 채울 땐 현재 row가 0으로 맨 윗줄이고, 현재 col이
aMagicSquare._order/2의 값으로 한 가운데 열이다.
값이 1인 cellValue를 선언해주고 보드의 현재 위치에 cellValue를 넣는다.
lastCellvalue는 aMagicSquare._order * aMagicSquare._order 를 값으로 가지고, cellValue는 2이다.
for문으로 cellValue가 2부터 (aMagicSquare._order *
aMagicSquare._order)까지 증가하며 코드를 반복한다.
nextLoc._row가 currentLoc_row-1 이고, nextLoc._row 가 0보다 작다면 nextLoc._row는 맨 밑줄 위치로 되고, nextLoc._col이 currentLoc._col+1 이고, nextLoc._col이 aMagicSquare._order보다 크거나 같으면 nextLoc._col은 가장 왼쪽 열 위치로 된다.
만약 다음 위치가 채워져 있으면, 바로 아래칸을 다음 위치로 수정한다.
다음 위치를 새로운 현재 위치로 하고, aBoard의 새로운 현재위치에 cellValue를 넣는다.

9. 풀이가 완료되면 AppIO_out_board(magicSquare._order,magicSquare._board)로 마방진 판을 화면에 보여준다.
10. AppIO_out_board(magicSquare._order,magicSquare._board)는 이중 for문을 사용해 [0],[1],[2] 등 마방진 판을 구성하는 프레임을 출력하는데,
주어진 차수까지 col과 row를 증가시키면서 출력한다. 그리고 aBoard에 저장된 row,col 값을 출력한다.

10. 마방진이 출력되면 다음 마방진을 위해 차수를 입력받는다,
11, 차수로 -10이 입력되면 마방진 풀이를 종료하고 AppIO_out_msg_endMagicSquare()으로 마방진 풀이 종료 메시지를 출력한다.

4. 실행 결과

```
<<<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 : 6
오류 : 차수가 짝수입니다. 차수는 홀수이어야 합니다
마방진 차수를 입력하시오 : 100
오류 : 차수가 너무 큽니다. 99보다 작아야 합니다.
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
```