2014 자료구조 실습과제 9

[실습 9] 스택의 응용 (미로찾기)

스택을 이용하여 미로탐색 프로그램 구현하시오.

- 단 스택은 배열로 구현해도 좋음 (linked-list가 아니어도 됨)
- 교재의 코드를 참고해서 구현하면 됨 (교재 208쪽 ~ 213쪽)
- 스택은 반드시 클래스로 구현할 것 (CHkdStack)
- 미로 클래스는 만들어 사용해도 되고 안해도 됨 (예: CHkdMaze)
- 적절한 main()함수를 만들어 동작을 확인할 수 있도록 함
- 입력은 반드시 파일로 받음 (자신만의 미로를 만들어 테스트 함)
- 임의의 크기의 미로를 처리할 수 있어야 함.
- 첨부의 소스 및 실행파일 참조 (실행 파일 및 데이터 파일(미로 파일))
- (보너스) 입구(5)에서 출구(9)으로 찾아가는 경로를 출력함

※ 반드시 클래스와 파일 이름들에 자신의 이니셜을 넣어서 구현하여야 함.
그렇지 않을 경우 감점처리 함.

※ main함수 예 (파일 첨부)

```
#include "Maze.h"

void main()
{
    CMaze   maze;

    // char str[200];
    // printf( " 미로 파일 이름을 입력하세요: ");
    // scanf("%s", str);
    // maze.Load(str);
    maze.Load("maze1.txt");
    maze.Print();
    printf( " 엔터를 치면 미로 입구 ○에서 출구 ◎로 가는 경로찾기를 시작합니다.\n");
    getchar();

    maze.searchExit();
    maze.Print();
    printf( " 과제는 일단 여기까지입니다. 다음은 보너스 항목입니다.\n\n");
    printf( " 엔터를 치면 탐색한 경로 □중에서 최적 경로 ☆를 출력합니다.\n");
    getchar();

    maze.PrintOptimal( );
    maze.Print();
    getchar();
}
```

※ Stack 클래스 예 (파일 첨부)

- Point클래스는 만들지 않아도 됨

```
struct CKdPoint {
    int     x;
    int     y;
    CKdPoint ( int xx=0, int yy=0 ) {
        x = xx;
        y = yy;
    }
    bool operator==( CKdPoint &p ) {
        return (p.x==x && p.y==y);
    }
    bool isNeighbor( CKdPoint &p ) {
        int dx = (x > p.x) ? (x - p.x) : (p.x - x);
        int dy = (y > p.y) ? (y - p.y) : (p.y - y);
        return ((dx+dy) == 1);
    }
};
class CStackPath
{
public:
    CKdPoint    m_path[10000];
    int         m_top;

    void push( int x, int y ) {
        push( CKdPoint( x, y ) );
    }
    void push( CKdPoint &pt ) {
        m_path[m_top++] = pt;
    }
    CKdPoint* pop () {
        if( m_top==0 ) return NULL;
        else  return &m_path[--m_top];
    }
    CKdPoint* peek () {
        if( m_top==0 ) return NULL;
        else  return &m_path[m_top-1];
    }

    CStackPath(void)    { m_top = 0; }
    ~CStackPath(void)   { }


    bool isEmpty() { return m_top==0 ; }
};
```

※ Maze 클래스 예 (파일(일부) 첨부)

- 클래스를 만들지 않고 main()에서 처리해도 되나, 클래스 사용을 권장함.

```
#pragma once
#include <stdio.h>
#include <stdlib.h>
#include "StackPath.h"
#define T_WALL  0
#define T_START 5
#define T_EXIT  9
#define T_EMPTY 1
#define T_DONE  2
#define T_OPTIMAL 8

class CMaze
{
public:
    // Member Data
    int     m_w;            // 미로의 width
    int     m_h;            // 미로의 height
    int**   m_elem;         // 미로의 각 항목 값을 저장
    CStackPath  m_stack;    // 미로 탐색을 위한 스택
    CKdPoint    m_start;    // 미로의 입구
    CKdPoint    m_exit;     // 미로의 출구

    CStackPath  m_optimal;  // 최적 경로 저장을 위한 스택

    CMaze(void);                    // 생성자
    ~CMaze(void);                   // 소멸자

    void    Init(int w, int h) {    // 이차원 배열 할당
        m_h = h;
        m_w = w;
        m_elem = new int* [ m_h ];
        for (int i=0 ; i<m_h ; i++ )
            m_elem[i] = new int[m_w];
    }

    void setBeginEnd();             // 입출구 설정 및 스택 초기화
    void searchExit();              // 미로찾기 시작
    void push(int x, int y);
    void addOptimalPath (CKdPoint &pt);

    bool isValidPos( int x, int y ) {   // 현재 위치가 미로 내인가?
        return (x>=0 && x<m_w && y>=0 && y<m_h);
    }
    bool isExitPos( int x, int y ) {    // 현재 위치가 출구인가?
        return (x==m_exit.x && y==m_exit.y);
    }
    int& get(int x, int y ) {           // 현재 위치의 값 반환
        return m_elem[y][x];
    }
    bool isPossiblePos( int x, int y ) {// 갈수 있는 길인가?
        return (get(x,y) == T_EMPTY);
    }
    void Load( char *fname = NULL );    // 파일/키보드 에서 Maze정보를 읽어옴
    void Print(char *fname = NULL );    // 현재 Maze를 파일/화면에 저장
    void PrintOptimal( );
};
```

※ Maze 클래스 구현 일부 (일부 코드 첨부)

- 입력 및 출력 함수

```
// 파일 입출력 함수
void CMaze::Load( char *fname ){     // 파일에서 리스트 정보를 읽어옴
    FILE *fp = stdin;
    if( fname != NULL ) {
        fp = fopen (fname, "r");
        if( fp == NULL ) {
            printf( " Error: %s 파일이 없습니다.\n");
            return;
        }
    }

    int w, h;
    printf( " 이미지 크기 입력 (w h): ");
    fscanf( fp, "%d%d", &w, &h );
    printf( " 이미지의 크기는 %d x %d 입니다.\n", w, h);

    Init( w, h );        // 메모리 할당

    for( int i=0 ; i<h ; i++ ) {
        for( int j=0 ; j<w ; j++ ) {
            fscanf( fp, "%d", &(m_elem[i][j]) );
        }
    }
    if( fp != stdin ) fclose(fp);
}

void CMaze::Print( char *fname ) {  // 현재 리스트를 파일에 저장
    FILE *fp = stdout;
    if( fname != NULL ) {
        fp = fopen (fname, "w");
        if( fp == NULL ) {
            printf( " Error: %s 파일을 만들 수 없습니다.\n");
            return;
        }
    }

    system("cls");
    printf( "============================================\n");
    printf( "  전체 미로의 크기 = ");
    fprintf( fp, "%d %d\n", m_w, m_h);
    printf( "============================================\n");

    for( int i=0 ; i<m_h ; i++ ) {
        for( int j=0 ; j<m_w ; j++ ) {
            switch (m_elem[i][j]) {
                case T_WALL     : fprintf( fp, "■" ); break;
                case T_START    : fprintf( fp, "○" ); break;
                case T_EXIT     : fprintf( fp, "◎" ); break;
                case T_EMPTY    : fprintf( fp, "  " ); break;
                case T_DONE     : fprintf( fp, "□" ); break;
                case T_OPTIMAL  : fprintf( fp, "☆" ); break;
            }
        }
        fprintf( fp, "\n");
    }
    printf( "============================================\n");
    if( fp != stdout ) fclose(fp);
}
```