HW6: stack

실습만 진행합니다.

1. makefile

명령어:

```
cat makefile / vi makefile
hw6:  hw6.o maze.o
      g++ -o hw6 hw6.o maze.o
```

또는 Visual Studio의 프로젝트 관리 기능을 사용합니다.

2. input file

(a) *m* by *p* 짜리 maze를 저장한 `maze.in`을 만드시오.

(첫 줄은 12 by 15 maze임을 나타내고, 그 후 12줄의 maze가 표기된다.)

명령어:

```
cat maze.in / vi maze.in
12  15
0 1 0 0 0 1 1 0 0 0 1 1 1 1 1
1 0 0 0 1 1 0 1 1 1 0 0 1 1 1
0 1 1 0 0 0 0 1 1 1 1 0 0 1 1
1 1 0 1 1 1 1 0 1 1 0 1 1 0 0
1 1 0 1 0 0 1 0 1 1 1 1 1 1 1
0 0 1 1 0 1 1 1 0 1 0 0 1 0 1
0 0 1 1 0 1 1 1 0 1 0 0 1 0 1
0 1 1 1 1 0 0 1 1 1 1 1 1 1 1
0 0 1 1 0 1 1 0 1 1 1 1 1 0 1
1 1 0 0 0 1 1 0 1 1 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0 1 1 1 1 0
0 1 0 0 1 1 1 1 1 0 1 1 1 1 0
```

(b) 9 by 9 짜리 maze를 저장한 `maze2.in` 도 만드시오.

명령어:

```
cat maze2.in / vi maze2.in
9 9
0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 0
1 0 0 0 0 0 0 0 1
0 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 0
1 0 0 0 0 0 0 0 1
0 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 0
```

3. maze 길찾기 프로그램 구현

(a) 다음 같이 동작하는 프로그램을 작성하려 한다.

자료 maze2.in에 대해서도 동작하는지 확인하라.

```
make hw4
hw4 maze.in
For maze datafile (maze.in)
 -> (1,1) -> (2,2) -> (1,3) -> (1,4) -> (1,5)
 -> (2,4) -> (3,5) -> (3,4) -> (4,3) -> (5,3)
 -> (6,2) -> (7,2) -> (8,1) -> (9,2) -> (10,3)
 -> (10,4) -> (9,5) -> (8,6) -> (8,7) -> (9,8)
 -> (10,8) -> (11,9) -> (11,10) -> (10,11) -> (10,12)
 -> (10,13) -> (9,14) -> (10,15) -> (11,15) -> (12,15)

#nodes visited = 48 out of 180
```

```
hw4 maze2.in
For maze datafile (maze2.in)
-> (1,1) -> (1,2) -> (1,3) -> (1,4) -> (1,5)
-> (1,6) -> (1,7) -> (1,8) -> (2,9) -> (3,8)
-> (3,7) -> (3,6) -> (3,5) -> (3,4) -> (3,3)
-> (3,2) -> (4,1) -> (5,2) -> (5,3) -> (5,4)
-> (5,5) -> (5,6) -> (5,7) -> (5,8) -> (6,9)
-> (7,8) -> (7,7) -> (7,6) -> (7,5) -> (7,4)
-> (7,3) -> (7,2) -> (8,1) -> (9,2) -> (9,3)
-> (9,4) -> (9,5) -> (9,6) -> (9,7) -> (9,8)
 -> (9,9)

#nodes visited = 40 out of 81
```

(b) main 프로그램(hw6.cpp)

```
#include <stdlib.h>

#include <fstream>
#include <iostream>
using namespace std;

void getdata(istream&, int&, int&);
void Path(int, int);
void ShortestPath(int, int);  // 조금 더 해보고 싶은 학생 작성

int main(int argc, char* argv[]) {
  int m, p;  // m by p maze
  if (argc == 1)
    cerr << "Usage: " << argv[0] << " maze_data_file" << endl;
  else {
    ifstream is(argv[1]);
    if (!is) {
      cerr << argv[1] << " does not exist\n";
      exit(1);
    }
    cout << "For maze datafile (" << argv[1] << ")\n";
    getdata(is, m, p);
    is.close();
    Path(m, p);
  }
}
```

(c) maze.cpp

```cpp
#include <iostream>
#include <stack>
using namespace std;
const int MAXSIZE = 100;  // up to 100 by 100 maze allowed
bool maze[MAXSIZE + 2][MAXSIZE + 2];
bool mark[MAXSIZE + 1][MAXSIZE + 1] = {0};

enum directions { N, NE, E, SE, S, SW, W, NW };
struct offsets {
  int a, b;
} move[8] = {{-1, 0}, {-1, 1}, {0, 1},  {1, 1},
             {1, 0},  {1, -1}, {0, -1}, {-1, -1}};

struct Items {
  Items(int xx = 0, int yy = 0, int dd = 0) : x(xx), y(yy), dir(dd) {}
  int x, y, dir;
};

template <class T>
ostream& operator<<(ostream& os, stack<T>& s) {
  // 스택의 내용을 역순으로 출력
  // 구현방법=내용을 하나씩 꺼내 다른 임시 스택에 넣어 저장한 후,
  // 최종적으로 그 임시 스택에서 하나씩 꺼내 출력하면 됨

  stack<T> temp;  // 역으로 출력하기 위해 임시 스택 temp 이용
  /*
    TODO: 이 부분 작성
  */
  return os;
}

ostream& operator<<(ostream& os, Items& item) {
  // 5 개의 Items 가 출력될 때마다 줄바꾸기위해
  static int count = 0;
  os << "(" << item.x << "," << item.y << ")";
  count++;
  if ((count % 5) == 0) cout << endl;
  return os;
}

void Path(const int m, const int p) {
  mark[1][1] = 1;
  stack<Items> stack;  // C++ STD stack 을 이용하기
  Items temp(1, 1, E);
  stack.push(temp);
```

```
  /* 최종적인 경로의 출력은 다음과 같이 한다.
    cout << stack;
    temp.x = i; temp.y = j; cout << " -> " << temp;
    temp.x = m; temp.y = p; cout << " -> " << temp << endl;
  */
}

void ShortestPath(const int m, const int p) {
  /*
  TODO: 더 실습해보고 싶은 학생들 작성
  다음 페이지의 '4. 생각해보기' 참고
  */

  /*
  hw6.cpp 의 main 함수도 Path 와 ShortestPath 둘 다 출력될 수
  있도록 수정 및 출력 결과 구분
  (e.g. Path 의 결과: ~~~ \n ShortestPath 의 결과: ~~~)
  */
}

void getdata(istream& is, int& m,
             int& p) {  // 자료화일을 읽어들여 maze 에 저장한다.
  is >> m >> p;
  for (int i = 0; i < m + 2; i++) {
    maze[i][0] = 1;
    maze[i][p + 1] = 1;
  }
  for (int j = 1; j <= p; j++) {
    maze[0][j] = 1;
    maze[m + 1][j] = 1;
  }
  for (int i = 1; i <= m; i++)
    for (int j = 1; j <= p; j++) is >> maze[i][j];
}
```

4. 생각해보기 (모든 학생 공통)

- stack을 사용하는 이유

- maze에서 올바른 경로를 찾아가는 과정

- 최단 경로 중 가장 적은 수의 노드를 거치는 경우를 찾고 싶을 때

(더 생각해보고 싶은 학생은 이것을 ShortestPath() 함수로 구현하면 됨)