과제5 보고서

B211103 변준석

2017년 10월

1 개요

이번 과제의 메인 프로그램인 *hw5.cpp* 에서는 스택 자료구조를 이용한 Maze의 경로 찾기 프로그램을 구현하였다.

2 문제 해결 방안

*maze*문제의 해결을 위하여 이동 방향을 결정하는 구조체*offsets*와 현재 위치 및 이동 방향을 값으로 갖는 구조체*Items*를 설계하였다. 경로를 찾는 함수는 *Path*로 구현하였다.

3 *최종 출력*

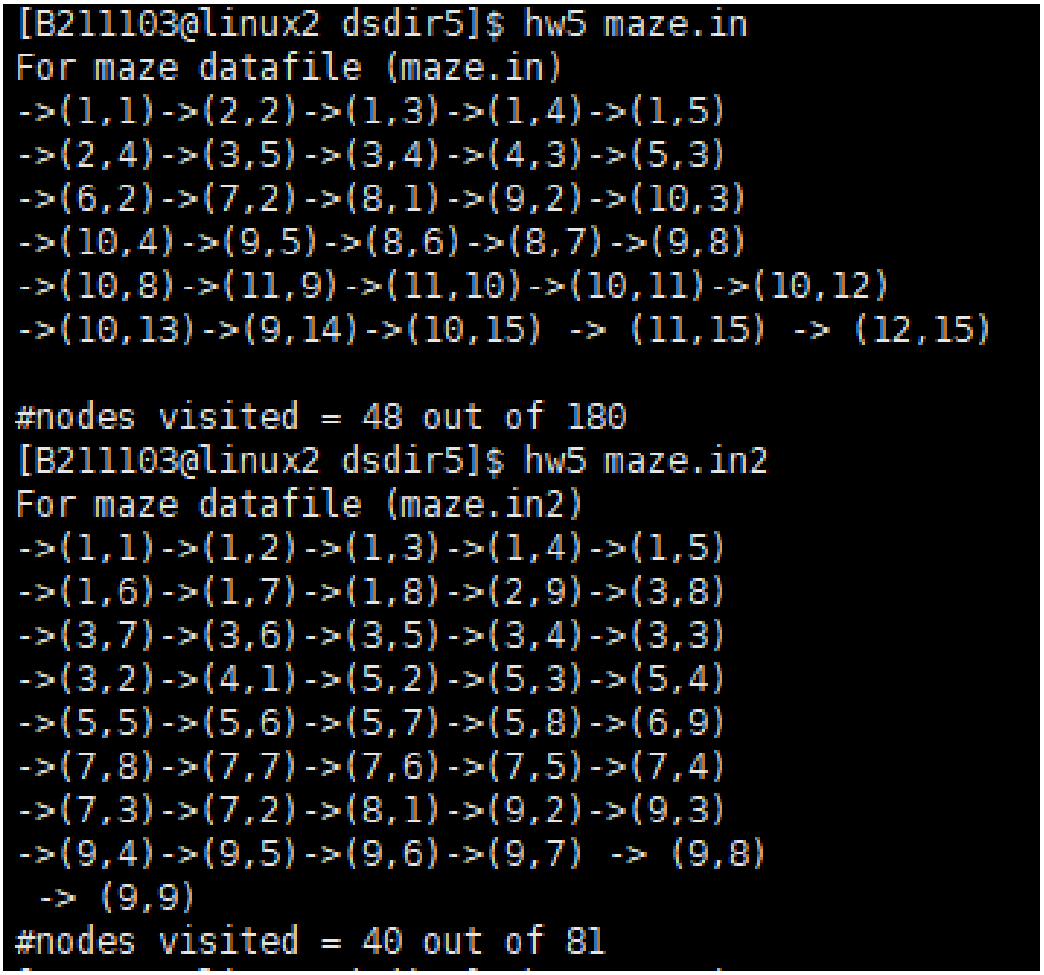

Figure 1: matrixa.cpp의 Transpose 구현

4 hw5의 간단한 설명

모든 기능들이 정상작동함을 알 수 있다. *nodes visited*변수는 스태틱 변수이며 *mark*프로세스마다 1씩 증가하였다.

```cpp
#include <iostream>
#include <fstream>
#include <stdlib.h>
using namespace std;

void getdata(istream&, int&, int&);      //maze data를 얻는 함수.
void Path(int, int);                     //경로 탐색을 하는 함수.

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);
   }
}
```

Figure 2: matrixa.cpp의 Transpose 구현

```cpp
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 offset move[8] = {{.a = -1, .b = 0}, {.a = -1, .b = 1}, {.a = 0, .b
//struct offset move[8] = {{-1, 0}, {-1, 1}, {0, 1}, {1, 1}, {1, 0}, {1, -1},

struct Items {
  Items(int xx=0, int yy=0,int dd=0): x(xx), y(yy), dir(dd) {}
  int x, y, dir;
}; //x,y,dir 값을 초기화하는 생성자를 갖는다.

template <class T>
ostream& operator<<(ostream& os, stack<T>& s) {
  static int count = 0;

  stack<T> s2;
  while(!s.empty()){
    s2.push(s.top());
    s.pop();
  }
  while(!s2.empty()){
    os << "->" << s2.top();
    s.push(s2.top());
    s2.pop();
  }
      // 스택의 내용을 역순으로 출력
      // 구현방법=내용을 하나씩 꺼내 다른 임시 스택에 넣어 저장한 후,
      // 최종적으로 그 임시 스택에서 하나씩 꺼내 출력하면 됨

  return os;
}
```

Figure 3: matrixa.cpp의 연산자 오버로딩

```
struct Items {
  Items(int xx=0, int yy=0,int dd=0): x(xx), y(yy), dir(dd) {}
  int x, y, dir;
}; //x,y,dir 값을 초기화하는 생성자를 갖는다.

template <class T>
ostream& operator<<(ostream& os, stack<T>& s) {
  static int count = 0;

  stack<T> s2;
  while(!s.empty()){
    s2.push(s.top());
    s.pop();
  }
  while(!s2.empty()){
    os << "->" << s2.top();
    s.push(s2.top());
    s2.pop();
  }
      // 스택의 내용을 역순으로 출력
      // 구현방법=내용을 하나씩 꺼내 다른 임시 스택에 넣어 저장한 후,
      // 최종적으로 그 임시 스택에서 하나씩 꺼내 출력하면 됨

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

Figure 4: matrixb.cpp의 Transpose 구현

```
void Path(const int m, const int p){
  int count = 0;
  //미로의 경로 탐색 함수.
  mark[1][1] = 1;
  stack<Items> stack;
  Items temp(1,1,E);
  stack.push(temp);

  while(!stack.empty()){
    temp = stack.top();
    stack.pop();
    int i = temp.x;
    int j = temp.y;
    int d = temp.dir;
    while (d < 8){
      int g = i + move[d].a;
      int h = j + move[d].b; //g와 h는 다음에 이동할 위치.
      if ((g==m)&&(h == p)){
        cout << stack;
        temp.x = i; temp.y = j; cout << " -> " << temp;
        temp.x = m; temp.y = p; cout << " -> " << temp << endl;
        for (int a = 1; a <= m; a++){
            for (int b = 1; b <= p; b++){
              if (mark[a][b] == 1) count++;
            }
       }
        cout << "#nodes visited = " << count << " out of " << m*p << endl;
        return;
      }
    if ((!maze[g][h]) && (!mark[g][h])){
      mark[g][h] = 1;
      temp.x = i;
      temp.y = j;
      temp.dir = d + 1;
      stack.push(temp);
      i = g;
      j = h;
      d = N; // move to (g,h)
      }
    else d++;
   }
  }
  cout << "No path in maze." << endl;
}
```

Figure 5: matrixb.cpp의 연산자 오버로딩

```
void getdata(istream& is, int& m, int & p){
  // 자료화일을 읽어들여 maze에 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];
}
```

Figure 6: matrixb.cpp의 연산자 오버로딩