STRUTTURE DATI e LABORATORIO II
Esercitazione n° 6

Il problema del labirinto

Supponi di avere un labirinto rappresentato da una matrice n x m i cui elementi sono 0 (zero) o 1 (uno) che indicano se in una posizione è possibile accedere (zero) oppure no (uno). Si richiede di costruire un programma in linguaggio C che determini il cammino che porta all'uscita del labirinto nel caso questo esista.

Suggerimenti:

1. Il cammino può essere costituito dalle posizioni sulla matrice che un ipotetico giocatore occupa nella matrice del labirinto

2. Per semplificare l'algoritmo si può supporre che la matrice del labirinto sia circondata da una barriera di 1; inoltre che l'uscita del labirinto sia nota e denotata da RIGA_ENTRATA e COL_ENTRATA;

3. Uno spostamento lungo una direzione può essere rappresentato dalla seguente struttura

```
typedef struct {
        int vert;
        int orizz;
  } offset;
```

4. Tutti i possibili spostamenti possono rappresentarsi col seguente array

```
offsetmossa[8]={{-1,0},  {-1,1},  {0,1}, {1,1},  {1,0}, {1,-1}, {0,-1}, {-1,-1}};
```

5. Per individuare la soluzione si può usare uno stack in cui viene memorizzato il punto corrente e lo spostamento da effettuare nel caso in cui lo spostamento attuale non porti all'uscita del labirinto.

   Lo stack potrebbe implementarsi con un array come segue:

```
typedef struct {
        int riga;
        int col;
        int dir;
    } elemento;

 elemento stack[MAX_STACK_SIZE];
```

   in cui MAX_STACK_SIZE è una costante simbolica.

6. La funzione di ricerca può essere

```
int path(void)
     /* genera un percorso attraverso un labirinto se tale percorso esiste*/
{
     void add(int *top, elemento item);
     elemento  delete(int *top);
     int riga, col,riga_succ, col_succ, dir, trovato=FALSE;
     elemento posizione;
```

```
segna[1][1]=1;    top=0;
stack[0].riga= RIGA_ENTRATA;
stack[0].col= COL_ENTRATA;
stack[0].dir=1;
while (top>-1 &&!trovato) {
      posizione=delete(&top);
      riga=posizione.riga;            col=posizione.col;
      dir=posizione.dir;
      while (dir<8 &&!trovato) {
                                                        /*mossa nella direzione dir */
          riga_succ=riga+mossa[dir].vert;
          col_succ=col+mossa[dir].orizz;
          if(riga_succ==RIGA_USCITA && col_succ==COL_USCITA)
                trovato=TRUE;
          else if(!lab[riga_succ][col_succ]  &&  !segna[riga_succ][col_succ]) {
                segna[riga_succ][col_succ]=1;
                posizione.riga=riga;posizione.col=col;
                posizione.dir=++dir;
                add(&top, posizione);
                riga=riga_succ;  col=col_succ;     dir=0;
               }
             else ++dir;
          }
   }
   return trovato;
}
```

in cui RIGA_ENTRATA e COL_ENTRATA indicano il punto di partenza e l'array segna[][] indica se un punto è stato raggiunto o non è stato raggiunto.

Le funzioni di inserimento e cancellazione nello stack sono

```
void add(int *top, elemento item)     /* aggiunge un elemento allo stack globale */
{
    if(*top >=MAX_STACK_SIZE-1) {
          printf("\nstack_full");
          return;
         }
 stack[++*top] = item;
}


elemento  delete(int *top)
  {
   if(*top == -1)
         printf("\nstack_empty");            /*fornisce un codice d'errore*/
   return stack[(*top)--];
```

7. Introduci una funzione per la visualizzazione del labirinto e la stampa del percorso trovato. Questo consiste nei punti memorizzati nello stack.

Considera come esempio la rappresentazione del seguente labirinto

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
0,1,1,1,1,0,0,1,1,1,1,1,1,1,1
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
0,0,1,1,1,1,1,1,0,0,0,1,1,1,0
0,1,0,0,1,1,1,1,1,1,0,1,1,1,0

che in C può rappresentarsi seguendo l'indicazione di introdurre una barriera di 1

```
int lab[MAX_RIGA][MAX_COL]={
                {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
                {1,0,1,0,0,0,1,1,0,0,0,1,1,1,1},
                {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},
                {1,1,1,0,1,1,1,1,0,1,1,0,1,0,0},
                {1,1,1,0,1,0,0,1,0,1,1,1,1,1,1},
                {1,0,0,1,1,0,1,1,1,0,1,0,0,1,0},
                {1,0,1,1,1,1,0,0,1,1,1,1,1,1,1},
                {1,0,0,1,1,0,1,1,0,1,1,1,1,0,1},
                {1,1,1,0,0,0,1,1,0,1,1,0,0,0,0},
                {1,0,0,1,1,1,1,1,1,0,0,0,1,1,0},
                {1,0,1,0,0,1,1,1,1,1,1,0,1,1,0},
                {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}};

int RIGA_USCITA=11, COL_USCITA=15;
int RIGA_ENTRATA=1, COL_ENTRATA=1;
```

Buon lavoro!