Estruturas de Dados e Algoritmos
Prof.: Fernando W Cruz

Exercício sobre labirinto

A figura 1 abaixo apresenta um labirinto, mapeado como uma matriz Labirinto[N][N], com posições livres (com valor zero) e paredes (posições com valor 32767).

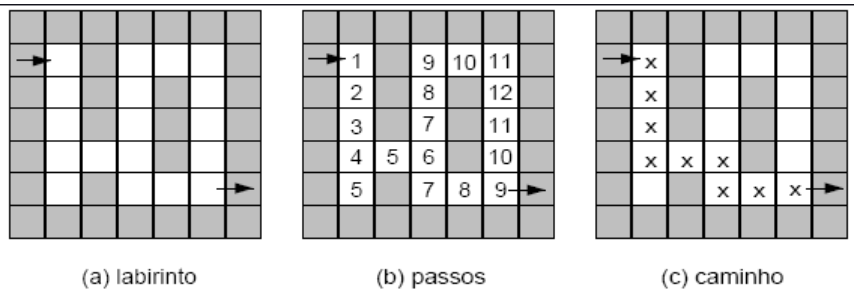

Figura 1 – Matriz bidimensional representando um labirinto.

Assumindo a entrada do labirinto na posição (1,1) e saída na posição (5,5) e que os movimentos não podem ser diagonais (somente horizontais e verticais), o aluno deve encontrar os caminhos que dão acesso à saída do labirinto e, em seguida, encontrar o menor caminho a ser seguido a partir de um dado ponto. Para isso, é necessária a utilização de dois grandes mecanismos:

- Filas - para realizar a anotação da matriz, com as possibilidades de caminhos. Nesse caso, é anotada na matriz L o número mínimo de passos necessários para atingir cada uma das posições do labirinto, a partir da entrada. Assim, anotação de que L[i][j]=k indica que são necessários k passos para atingir a posição (i,j), partir da posição (1,1), conforme ilustrado na Figura 1-b. Caso uma posição qualquer da matriz L permaneça com valor 0, após a fase de anotação, isto significa que não existe um caminho que leve da entrada até esta posição. Particularmente, se a posição (N-2,N-2) permanece zerada após a anotação da matriz, isto significa que o labirinto não tem saída.
- Pilhas - após identificar do menor caminho pelo processo de anotação, é preciso marcar esse caminho, colorindo ou inserindo algum outro identificador no percurso identificado. Para essa funcionalidade, pode-se utilizar o recurso de pilhas, como será visto mais adiante, no exercício c.

Com base nessas informações, resolva o que se pede a seguir

a) **Digitar e testar o programa dado abaixo.**

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#define MAX 100 // capacidade de armazenamento da pilha e da fila
typedef int titem; // tipo dos itens armazenados na pilha e na fila
#define N 22 // tamanho da matriz que representa o labirinto
#define LIVRE 0 // marca de posicao livre no labirinto
#define PAREDE 32767 // marca de posicao com parede no labirinto

void cria(int L[N][N]) { // funcao para criacao de um labirinto
     int i, j;
     for(i=0; i<N; i++) {
      L[i][0] = PAREDE;
      L[i][N-1] = PAREDE;
     }
     for(j=0; j<N; j++) {
      L[0][j] = PAREDE;
      L[N-1][j] = PAREDE;
     }

     for(i=1; i<N-1; i++)
      for(j=1; j<N-1; j++)
            if( rand()%3==0 ) L[i][j] = PAREDE;
            else L[i][j] = LIVRE;
     L[1][1] = LIVRE;
     L[N-2][N-2] = LIVRE;
} /* fim-cria */

void exibe(int L[N][N]) { // funcao para exibicao de um labirinto
     int i, j;
     for(i=0; i<N; i++) {
      for(j=0; j<N; j++)
            switch( L[i][j] ) {
                  case LIVRE : putchar(' '); break;
                  case PAREDE: putchar('#'); break;
                  default : putchar(126);
            }
      printf("\n");
     }
}
//Adicione aqui o codigo da função anota() - exercício 2
//Adicione aqui o codigo da função extrai() - exercício 3
int main(void) { // Função principal
     int L[N][N];
     char r;
     srand(time(NULL));
     do {
      system ("cls");
      cria(L);
      /* anota(L); */ // retire o comentario na versão final
      /* extrai(L); */ // retire o comentario na versão final
      exibe(L); // retire esta chamada na versão final
//    _gotoxy(1,N+3); /* util se utilizar a lib conio.h ou similar */
      printf("Continua? (s/n) ");
      scanf("%c%*c",&r);
     } while( toupper(r)!='N' );
     return 0;
} /* fim-main */
```

b) **Elaborar a função anota(), considerando o algoritmo a seguir:**

1. Armazene o valor 1 na posição L[1][1];
2. Insira a posição (1,1) numa fila vazia (vetor fila);
3. Enquanto a fila não estiver vazia, faça:
    3.1 Remova uma posição (i,j) da fila F;
    3.2 Defina c igual a L[i][j]+1
    3.3 Para cada posição (x,y) vizinha de (i,j) tal que L[x][y]==0 faça:
        3.3.1 Armazene o valor c em L[x][y];
        3.3.2 Insira a posição (x,y) na fila F.

c) Escreva a função extrai() que é responsável por apresentar o menor caminho a partir da matriz anotada (função anota()), como exemplificado na Figura 1-c. Para a construção dessa função, considerar o seguinte algoritmo:

1. Exiba o labirinto L no vídeo;
2. Se a posição L[N-2][N-2] estiver zerada, informe que não há saída e pare a execução;
3. Insira a posição (N-2, N-2) numa pilha vazia P;
4. Enquanto a posição (1,1) não estiver no topo da pilha P, faça:
    4.1 Acesse, sem remover, a posição (i,j) armazenada no topo de P;
    4.2 Encontre uma posição (x,y) vizinha de (i,j) tal que L[x][y]==L[i][j]-1;
    4.3 Insira a posição (x,y) na pilha P;
5. Enquanto a pilha P não estiver vazia, faça:
    5.1 Retire uma posição (i,j) da pilha P;
    5.2 Posicione o cursor na linha i+1 e coluna j+1 do vídeo;
    5.3 Exiba o caractere 'x'