Alvarado Sandoval Alberto 3CV14

Problema de los laberintos

Diseñé una implementación que genera laberintos aleatorios en una matriz nxn y los soluciona.

Código:

//Algoritmo para encontrar camino en laberinto

//Autor: Alberto "El Rifadito" Alvarado

//Fecha: 17-Mar-2021

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<unistd.h>

**typedef** **struct** paso{

**int** x, y;

**struct** paso \*ant;

**struct** paso \*sig;

}t\_paso, \*Paso;

**typedef** **struct** camino{

Paso inicial;

Paso final;

**int** longitud;

}t\_camino, \*Camino;

**int** **menu**(**void**);

**void** **imprimirLaberinto**(**int**\*\*, **int**);

**int**\*\* **generarLaberinto**(**int**);

Camino **newCamino**(Paso);

Paso **newPaso**(**int**, **int**);

**int** **addPaso**(Camino c, Paso p);

**int** **encontrarCamino**(**int**\*\* lab, **int** n, Camino c, Paso p);

**int** **marcarCamino**(**int**\*\* lab, Camino c);

**int** **imprimirCamino**(Camino c);

**int**\*\* **imprimirSolucion**(Camino c, **int**\*\* lab);

**int** **main**(**void**){

menu();

}

**int** **menu**(**void**){

system("cls");

printf("\t\t\nSALIDA DEL LABERINTO\n\n");

**int** \*\*lab = **NULL**;

**int** N = 16, band;

Paso ini = **NULL**;

Camino c = **NULL**;

ini = newPaso(0, 0);

c = newCamino(ini);

lab = generarLaberinto(N);

band = encontrarCamino(lab, N, c, ini);

**if**(band == 1){

imprimirLaberinto(lab, N);

printf("\n\n\tCamino Encontrado!\n\n");

imprimirCamino(c);

printf("\n");

lab = imprimirSolucion(c, lab);

printf("\n");

imprimirLaberinto(lab, N);

}

**else**{

imprimirLaberinto(lab, N);

printf("\n\n\tLaberinto sin Salida.");

menu();

}

}

Paso **newPaso**(**int** a, **int** b){

Paso p = **NULL**;

p = (Paso) malloc(**sizeof**(t\_paso));

p->x = a;

p->y = b;

p->ant = **NULL**;

p->sig = **NULL**;

**return** p;

}

Camino **newCamino**(Paso ini){

Camino c = **NULL**;

c = (Camino)malloc(**sizeof**(t\_camino));

c->inicial = ini;

c->final = ini;

c->longitud = 1;

}

**void** **imprimirLaberinto**(**int** \*\*lab, **int** n){

**int** i=0, j = 0, bandera = 0;

**char** c;

printf("entrada");

printf("\t");

**for**(i = 0; i<(n+2); i++)

printf("%c",196);

printf("\n");

**for**(i=0; i<n; i++){

printf("\t%c", 179);

**for**(j = 0; j<n; j++){

**if**(lab[i][j] == 0)

c = 32;

**if**(lab[i][j] == 1)

c = 219;

**if**(lab[i][j] == 2)

c = 250;

printf("%c",c);

**if**(j == n-1)

printf("%c", 179);

}

printf("\n");

}

printf("\t");

**for**(i = 0; i<n; i++)

printf("%c",196);

printf(" salida");

}

**int**\*\* **generarLaberinto**(**int** n){

**int** \*\*lab = **NULL**;

**int** i, j, band=0, global = 0, alternador=0, sw=0;

srand (time(**NULL**));

lab = (**int**\*\*)malloc(n \* **sizeof**(**int**\*));

**for**(i = 0; i<n; i++)

lab[i] = (**int**\*)malloc(n \* **sizeof**(**int**));

**for**(i = 0; i<n; i++){

**for**(j = 0; j<n; j++){

**if**(i == 0 && j == 0 || i == n-1 && j == n-1 ){

lab[i][j] = 0;

global = n/2-4;

}

**else**{

**if**(global < n/2-1)

lab[i][j] = (**int**)rand()%2;

**else**

lab[i][j] = 0;

**if**(band>2\*n/-1)

alternador = 0;

**if**(alternador == 0)

**if**(i !=0 && j!=0 && i<n-1 && lab[i-1][j]==0 && band <3\*n/4){

lab[i][j] = 0;

alternador = 1;

band++;

}

**if**(alternador == 1){

**if**(lab[i][j-1] == 0 && lab[i-1][j-1] && band < 2\*n/3)

lab[i][j] = 0;

band++;

alternador = 0;

}

**if**(i < 2 && j<3)

lab[i][j]=0;

**if**(i<n/2+2 && i>n/2-2 && j>n+3 && j<n/2 && (lab[i][j-1] == 0 || lab[i-1][j] == 0)&& sw < n/2 ){

lab[i][j] = 0;

lab[i-1][j] = 0;

sw++;

}

**if**((i>n/4-2 && i<n/4+3 || (i>3\*n/4 && i<n-2) && j>n+4 && j<n/2+3 ) && (lab[i][j-1] == 0 || lab[i-1][j] == 0) && n>16 && sw<n){

lab[i][j] = 0;

sw++;

}

**if**(j>n/2+2 && lab[i][j-1] == 0 && lab[i][j-2] == 0 && lab[i][j-3] == 0 && lab[i][j-4] == 0 && (i<n/2 || (i>3\*n/4 && i<n-2)) && n>8)

lab[i][j] = 1;

**if**(j>n-3 && i<n-4&&(lab[i][j-1] == 0 || lab[i][j-2] == 0) ){

lab[i][j] = 1;

}

**if**(lab[i][j] == 1){

global++;

}

}

}

**if**(i<2 || (i>n-3 && i<n))

global = n/2-4;

**else**

global = i/4;

sw = 0;

}

**return** lab;

}

**int** **addPaso**(Camino c, Paso p){

Paso ant = **NULL**;

ant = c->final;

p->ant = ant;

ant->sig = p;

c->final = p;

c->longitud++;

**return** 1;

}

**int** **pop**(Camino c){

Paso p, f;

p = c->inicial;

f = c->final;

**int** i=0;

**while**(i< c->longitud-1){

p = p->sig;

i++;

}

p->sig = **NULL**;

c->final = p;

free(f);

}

**int** **encontrarCamino**(**int**\*\* lab, **int** n, Camino c, Paso p){

**int** x,y;

x = p->x;

y = p->y;

Paso sig;

**if**(x == n-1 && y == n-1){

addPaso(c, p);

**return** 1;

}

**if**(x<n && y<n && lab[y][x]!=1){

sig = newPaso(x+1,y);

**if**(encontrarCamino(lab, n, c, sig)){

addPaso(c, p);

**return** 1;

}

sig = newPaso(x,y+1);

**if**(encontrarCamino(lab, n, c, sig)){

addPaso(c, p);

**return** 1;

}

**return** 0;

}

**return** 0;

}

**int** **imprimirCamino**(Camino c){

Paso act, sig;

**int** i=0, x,y;

act = c->final;

printf("\n\nCamino:\n");

printf("\n");

**while**(i < c->longitud-1){

sig = act->ant;

printf("(%d, %d)",act->x, act->y);

i++;

**if**(i%5 == 0)

printf("\n");

**if**(i<c->longitud-1)

printf("->");

act = sig;

}

}

**int**\*\* **imprimirSolucion**(Camino c, **int**\*\* a){

**int** x,y, i=0;

Paso p, sig;

**int** \*\*lab;

lab = a;

p = c->final;

**while**(i<c->longitud -1){

sig = p->ant;

x = p->x;

y = p->y;

lab[y][x] = 2;

i++;

p = sig;

}

**return** lab;

}

Ejecución:



