Nome: João Paulo de Oliveira 11611BCC046

Labirinto

Uberlândia

2016

1.Código fonte:

- Labirinto.c:

```
#include<stdio.h>
#include <stdlib.h>
#include "labirinto.h"
#include <time.h>
int** cria_matriz(int tamanho){
   int** M,i,j;
   M = malloc(tamanho*sizeof(int*));
   if(M==NULL){
      puts("Memoria insuficiente!!!\n");
      exit(1);
   }
   for (i=0;i<tamanho;i++){
      M[i] = malloc(tamanho*sizeof(int));
      if(M==NULL){
      puts("Memoria insuficiente!!!\n");
      exit(1);
   }
      for (j=0;j<tamanho;j++){
         M[i][j]=9;
      }
   }
   return M;
}

void printa(int** matriz,int tamanho){
```

```
    int i,j;
    for (i=0;i<tamanho;i++){
        for (j=0;j<tamanho;j++){
            if(matriz[i][j]==9)
                printf(" - ");
            else printf(" %d ",matriz[i][j]);
        }
        puts("");
    }
    puts("");
}

void libera_matriz(int** M, int tamanho){
    int i;
    for(i=0;i<tamanho;i++)
        free(M[i]);
    free(M);
}

unsigned int conta01(int** mat, int i, int j, int tamanho){
    int count=0;
    if(i-1>=0     &&(mat[i-1][j]==0||mat[i-1][j]==1))count++;
    if(i+1<tamanho&&(mat[i+1][j]==0||mat[i+1][j]==1))count++;
    if(j-1>=0     &&(mat[i][j-1]==0||mat[i][j-1]==1))count++;
    if(j+1<tamanho&&(mat[i][j+1]==0||mat[i][j+1]==1))count++;
    return count;
}
```

```
int** cria_caminho(int** matriz,int i,int j,int tamanho,int min){
    int inci,incj,move=0;
    matriz[i][j]=1;
    srand(time(NULL));
    while(i>=0||j>=0||i<tamanho||j<tamanho){
          do{
            inci = rand()%3-1;
            incj = rand()%3-1;
          }while(inci==incj||inci==incj*-1);
          if(i+inci<0||j+incj<0||i+inci==tamanho||j+incj==tamanho){
            if(move>=min){
                break;
            }else continue;
          }
          if(matriz[i+inci][j+incj]==1||conta01(matriz,i+inci,j+incj,tamanho)>1) continue;
          i += inci;
          j += incj;
          matriz[i][j]=0;
          move++;
    }
    matriz[i][j]=2;
    return matriz;
}

void waitFor (unsigned int secs){
    unsigned int retTime = time(0) + secs;   // Get finishing time.
```

```
    while (time(0) < retTime);              // Loop until it arrives.
}

int do_lado_da_saida(int** mat,int i,int j,int tamanho){
    int count=0;
    if(i-1>=0&&mat[i-1][j]==2)count++;
    if(j-1>=0&&mat[i][j-1]==2)count++;
    if(i+1<tamanho&&mat[i+1][j]==2)count++;
    if(j+1<tamanho&&mat[i][j+1]==2)count++;
    return count > 0;
}

void cria_becos(int*** matriz ,const int tamanho){
    int i,j,ct=0,ct_d=0;
    srand(time(NULL));
    for(i=0;i<tamanho;i++){
        for(j=0;j<tamanho;j++){
            if(conta01((*matriz),i,j,tamanho)==1)
                ct++;
        }
    }
    if (ct>(tamanho*tamanho))ct = (tamanho*tamanho);
    ct-=(int)tamanho/5;
    while(ct_d<ct){
        i = rand()%tamanho;
        j = rand()%tamanho;
        if (conta01((*matriz),i,j,tamanho)!=1) continue;
```

```
        if ((*matriz)[i][j]==2) continue;
        if(do_lado_da_saida((*matriz),i,j,tamanho))continue;
        if((*matriz)[i][j]!=1){
                (*matriz)[i][j]=0;
                ct_d++;
        }
    }
}

void cria_labirinto(int*** matriz,const int tamanho){
    int min = tamanho/2,i,j;
    srand(time(NULL));
    do{
        puts("Digite um ponto de partida na borda valido:");
        scanf("%d %d",&i,&j);
    } while(i<0||j>=tamanho||j<0||i>=tamanho);
    (*matriz) = cria_caminho((*matriz),i,j,tamanho,min);
    cria_becos(matriz,tamanho);
}

pilha* cria_pilha(){
    pilha* stack;
    stack = malloc(sizeof(pilha));
    stack ->size=0;
    stack ->topo=NULL;
    return (stack);
}
```

```
void remove_pilha(pilha* p){
    pos* aux;
    aux = p->topo->prox;
    free(p->topo);
    p->topo = aux;
}

void libera_pilha(pilha* p){
    pos* aux;
    while(p->topo!=NULL){
        aux = p->topo->prox;
        free(p->topo);
        p->topo = aux;
    }
    free(p);
}

void insere_pilha(pilha* stack, int i, int j){
    pos *p;
    p = malloc(sizeof(pos));
    p->i=i;
    p->j=j;
    p->prox = stack->topo;
    stack->topo = p;
    (stack->size)++;
}
```

```
unsigned int redundancia(int** mat, int i, int j, int tamanho,int elemento){
    int count=0;
    if(i-1>=0&&mat[i-1][j]==elemento)count++;
    if(j-1>=0&&mat[i][j-1]==elemento)count++;
    if(i+1<tamanho&&mat[i+1][j]==elemento)count++;
    if(j+1<tamanho&&mat[i][j+1]==elemento)count++;
    return count;
}

void percorre (int*** matriz,int tamanho){
    int i=0,j=0;
    pilha* p = cria_pilha();
    for(i=0;i<tamanho;i++){
        for(j=0;j<tamanho;j++){
            if(((*matriz)[i][j])==1)break;
        }
        if(((*matriz)[i][j])==1)break;
    }
    do{
        if(redundancia((*matriz),i,j,tamanho,0)>1){
            puts("a");
            if(i-1>=0&&(*matriz)[i-1][j]==0){
                insere_pilha(p,i-1,j);
            }
            if(j-1>=0&&(*matriz)[i][j-1]==0){
                insere_pilha(p,i,j-1);
```

```
    }
    if(i+1<tamanho&&(*matriz)[i+1][j]==0){
        insere_pilha(p,i+1,j);
    }
    if(j+1<tamanho&&(*matriz)[i][j+1]==0){
        insere_pilha(p,i,j+1);
    }
    (*matriz)[i][j]=1;
    if(redundancia((*matriz),i,j,tamanho,2)!=0)break;
    i = p->topo->i;
    j = p->topo->j;
    remove_pilha(p);
    (*matriz)[i][j]=1;
    waitFor(1);
    system("cls");
    printa((*matriz),tamanho);
    continue;
}
if(redundancia((*matriz),i,j,tamanho,0)==1){
    if(i-1>=0     &&(*matriz)[i-1][j]==0)i--;
    else
        if(j-1>=0     &&(*matriz)[i][j-1]==0)j--;
        else
            if(i+1<tamanho&&(*matriz)[i+1][j]==0)i++;
            else
                if(j+1<tamanho&&(*matriz)[i][j+1]==0)j++;
    (*matriz)[i][j]=1;
```

```
        waitFor(1);
        system("cls");
        printa((*matriz),tamanho);
     }
    if(redundancia((*matriz),i,j,tamanho,0)==0){
        if(redundancia((*matriz),i,j,tamanho,2)!=0)break;
        (*matriz)[i][j]=1;
        i = p->topo->i;
        j = p->topo->j;
        (*matriz)[i][j]=1;
        remove_pilha(p);
        waitFor(1);
        system("cls");
        printa((*matriz),tamanho);
    }
    waitFor(1);
    system("cls");
    printa((*matriz),tamanho);
}while(redundancia((*matriz),i,j,tamanho,2)==0);
if(i-1>=0     &&(*matriz)[i-1][j]==2)(*matriz)[i-1][j]=1;
else
    if(j-1>=0     &&(*matriz)[i][j-1]==2)(*matriz)[i][j-1]=1;
    else
        if(i+1<tamanho&&(*matriz)[i+1][j]==2)(*matriz)[i+1][j]=1;
        else
            if(j+1<tamanho&&(*matriz)[i][j+1]==2)(*matriz)[i][j+1]=1;
libera_pilha(p);
```

```
    waitFor(1);
    system("cls");
}
```

- Labirinto.c:

```
#ifndef LABIRINTO_H_INCLUDED
#define LABIRINTO_H_INCLUDED
#define MIN 4

typedef struct posicao {
    int i;
    int j;
    struct posicao* prox;
}pos;

typedef struct {
    int size;
    pos* topo;
}pilha;
void percorre (int*** matriz,int tamanho);
int** cria_matriz(int tamanho);
void libera_matriz(int** M, int tamanho);
void printa(int** matriz,int tamanho);
void cria_labirinto(int*** matriz,const int tamanho);
void waitFor (unsigned int secs);
#endif // LABIRINTO_H_INCLUDED
```

- Main.c:

```
#include <stdio.h>
#include <stdlib.h>
#include "labirinto.h"
#include <ctype.h>
#include <string.h>
int main(){
    int tamanho;
    char opcao;
    while(1){

printf("**********************************************************************
**************\n");
        printf("*                                                    *\n");
        printf("*              Bem Vindo Ao Labirinto do AssusTAdo                  *\n");
        printf("*                                                    *\n");
        printf("*                                                    *\n");
        printf("*           Sou um fodendo labirinto que se resove sozinho e          *\n");
        printf("*                    sem a ajuda de humanos                    *\n");

printf("**********************************************************************
**************\n\n");
        puts("Digite:");
        puts("1 - Jogar");
        puts("2 - Sair");
        scanf("%c",&opcao);
        switch(opcao){
            case '1':
                system("cls");
```

```
            do{
               puts("Digite o tamanho do labirinto  >=1 :");
               scanf("%d",&tamanho);
            }while(tamanho<=1);
            system("cls");
            puts("\tLabirinto:");
            int **matriz = cria_matriz(tamanho);
            cria_labirinto(&matriz,tamanho);
            system("cls");
            percorre(&matriz,tamanho);
            puts("\tLabirinto Resolvido:");
            printa(matriz,tamanho);
            system("pause");
            libera_matriz(matriz,tamanho);
         break;

         case '2':
            exit(1);
         break;
         default:
            system("cls");
            continue;
         break;
      }
   }
   return 0;
}
```