Stephanie Wyckoff  
CSc 3320 Programming Challenge #11  
Account: swyckoff1  
Due date: 10/11/16

[GSUAD\swyckoff1@snowball ~]$ cat pc11\_main.c

/\*

\* pc11.c

\*

\* Stephanie Wyckoff

\* Account: swyckoff1

\* CSc 3320

\* Programming Challenge #11

\* Due date: 11/8/2016

\*

\* Discription: This program includes path.h which holds the functions to create a 12x12 matrix and randomly create a path of letters through the matrix until the path is blocked on all sides.

\*

\* Input: There is no input for this program.

\*

\* Output: The program generates the final path of letters through the matrix via path.h header file.

\*

\*/

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include "path.h"

#define ROWS 12

#define COLS 12

main(int argc, char \*argv[]) {

int dir=0;

char grid[ROWS][COLS];

char path[52] = {'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};

int letter = 0;

createMatrix(grid);

createPath(grid, path, dir, letter);

printFinalGrid(grid);

return 0;

}

[GSUAD\swyckoff1@snowball ~]$ cat path.h

/\*

\* path.h

\*

\* Stephanie Wyckoff

\* Account: swyckoff1

\* CSc 3320

\* Programming challenge #11, header file

\* Due date: 11/8/2016

\*

\* Discription: This program randomly generates paths through a 12x12 matrix using the letters of the alphabet from A-Z, a-z. A random number is chosen and modulo by 4 to provide the numerical value of the directions 0 = up, 1 = right, 2 = down, 3 = left. If the path is blocked by a letter already in the direction chosen, the program will find another direction for it to go in. If all paths are blocked, the program ends.

\*

\* Input: This program takes in parameters for the createMatrix(), createPath() and printFinalGrid() functions from a main program.

\*

\* Output: The program generates the final path of letters through the matrix.

\*

\*/

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#define ROWS 12

#define COLS 12

void createMatrix(char grid[ROWS][COLS]){

int row; int col;

for(row=0; row < ROWS; row++){

for(col=0; col < COLS; col++){

grid[row][col] = '.';

}

}

}

void createPath(char grid[ROWS][COLS], char path[52], int dir, int letter){

srand(time(NULL));

int row=0; int col=0;

grid[row][col] = path[letter];

letter++;

while(letter < 52){

dir = rand() % 4;

switch(dir){

case 0:

if(grid[row-1][col] == '.' && row !=0){

grid[row-1][col] = path[letter];

row--;

}

if(grid[row-1][col] != '.' && grid[row][col+1] != '.' && grid[row+1][col] != '.' && grid[row][col-1] != '.'){

letter = 52;

break;

}

case 1://checks one space right for open space

if(grid[row][col+1] == '.' && col != COLS-1){

grid[row][col+1] = path[letter];//assigns letter to space

letter++; col++;//updates grid positon

}

if(grid[row-1][col] != '.' && grid[row][col+1] != '.' && grid[row+1][col] != '.' && grid[row][col-1] != '.'){

letter = 52;

break;

}

case 2://checks one space down for open space

if(grid[row+1][col] == '.' && row != ROWS-1){

grid[row+1][col] = path[letter];//assigns letter to space

letter++; row++;//updates grid position

}

if(grid[row-1][col] != '.' && grid[row][col+1] != '.' && grid[row+1][col] != '.' && grid[row][col-1] != '.'){

letter = 52;

break;

}

case 3://checks one space left for open space

if(grid[row][col-1] == '.' && col != 0){

grid[row][col-1] = path[letter];//assigns letter to space

letter++; col--;//updates grid position

}

if(grid[row-1][col] != '.' && grid[row][col+1] != '.' && grid[row+1][col] != '.' && grid[row][col-1] != '.'){

letter = 52;

break;

}

}

}

}

void printFinalGrid(char grid[ROWS][COLS]){

int row; int col;

//prints the final grid with all letters of the path

for(row = 0; row < ROWS; row++){

for(col = 0; col < COLS; col++){

printf("%2c", grid[row][col]);

}printf("\n");

}

}