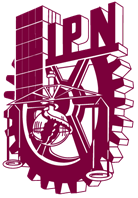
****

**INSTITUTO POLITÉNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**ESTRUCTURAS DE DATOS**

**PRÁCTICA #5**

**PROBLEMA DE LAS N-REINAS**

**GRUPO: 1CM8**

**EQUIPO: LAS MÁS PERRAS**

**INTEGRANTES:**

* JIMÉNEZ DELGADO LUIS DIEGO 2019630461
* SÁNCHEZ CASTRO AARÓN GAMALIEL 2019630079
* SÁNCHEZ TIRADO CITLALI YASMÍN 2019630096



**PROFESOR:** EDGARDO ADRIÁN FRANCO MARTÍNEZ

**FECHA DE ENTREGA:** 13 DE MAYO 2019

**INTRODUCCIÓN**

**MARCO TEÓRICO**

.

**DISEÑO Y FUNCIONAMIENTO DE LA SOLUCIÓN**

**IMPLEMENTACIÓN DE LA SOLUCIÓN**

**/\***

BACKTRACKING N-REINAS

AUTORES:

JIMÉNEZ DELGADO LUIS DIEGO

SÁNCHEZ CASTRO AARÓN GAMALIEL

SÁNCHEZ TIRADO CITLALI YASMÍN

VERSIÓN 1.5

DESCRIPCIÓN: EL PROGRAMA COLOCA N NÚMERO DE REINAS DADAS POR EL USUARIO EN UN TABLERO DE DIMENSIÓN NxN

EL OBJETIVO ES POSICIONAR A LAS REINAS DE MANERA QUE NINGUNA SE AMENACE CON EL RESTO.

\*/

#include <stdio.h>

#include <stdlib.h>

#include "Gotoxy.h"

//CONSTANTES A UTILIZAR

#define TRUE 1

#define FALSE 0

//VARIABLES GLOBALES POR COMODIDAD

int i**,**j**,**x**,**y**,**aux**;**

/\*

FUNCIÓN: validarPosicion(int fila, int posReinas[], int n)

RECIBE: int fila (FILA DE LA REINA A VALIDAR), int[] posReinas (POSICIONES DE LAS REINAS EN EL TABLERO)

int n (NÚMERO DE REINAS INGRESADO POR EL USUARIO)

DEVUELVE: TRUE/FALSE SEGÚN SEA EL CASO.

DESCRIPCIÓN: LA FUNCIÓN VALIDA QUE LA REINA ESTÉ BIEN COLOCADA, SE COMPRUEBA QUE PARA CUALQUIER FILA i

UNA REINA NO ESTÉ EN LA MISMA COLUMNA (SE COMPRUEBA QUE TODOS LOS NÚMEROS EN EL ARREGLO SENA DIFERENTES).

ADEMÁS, EL PROBLEMA DE LAS DIAGONALES ASCENDENTES/DESCENDENTES SE ARREGLA REALIZANDO LA OPERACIÓN

fila-columna, ESTA OPERACIÓN DEBE TENER UN RESULTADO DIFERENTE PARA CADA PAR ORDENADO DEL ARREGLO.

OBSERVACIONES:

\*/

int validarPosicion**(**int fila**,** int posReinas**[],** int n**)** **{**

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

//SI posReinas[i]==posReinas[fila] ENTONCES ESTÁN EN LA MISMA COLUMNA

//SI posReinas[fila]-posReinas[i]=(fila-columna) ENTONCES ESTÁN EN LA MISMA DIAGONAL

**if** **((**posReinas**[**i**]==**posReinas**[**fila**])||(**abs**(**fila**-**i**)** **==** abs**(**posReinas**[**fila**]-**posReinas**[**i**]))){**

**return** FALSE**;**

**}**

**}**

**return** TRUE**;**

**}**

La función *validarPosicion* recibe la fila en la que se ubica la reina agregada, mediante una sentencia for para recorrer el arreglo de posiciones se comprueba si dicha fila es una posición válida según la condición establecida en el problema.

/\*

FUNCIÓN: mostrarTablero( int posReinas[], int n, int \*alto)

RECIBE: int[] posReinas (POSICIONES DE LAS REINAS EN EL TABLERO)

int n (NÚMERO DE REINAS INGRESADO POR EL USUARIO), int\* alto (ORDENADA PARA SITUAR EL CURSOR E IMPRIMIR)

DEVUELVE: NADA.

DESCRIPCIÓN: IMPRIME EL TABLERO CON LAS POSICIONES ALMACENADAS EN EL ARREGLO AL MOMENTO.

OBSERVACIONES: PARA CASOS DONDE n (NÚMERO DE REINAS) ES "GRANDE" (P.E: 10) ES NECESARIO QUE EL TAMAÑO DEL BÚFER

DE PANTALLA SEA IGUALMENTE GRANDE PARA EVITAR UN DESBORDAMIENTO Y LA PÉRDIDA DE LOS GRÁFICOS.

\*/

void mostrarTablero**(**int posReinas**[],** int n**,** int **\***alto**)**

**{**

x**=**0**;**

y**=\***alto**;**

aux**=**0**;**

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

x**=**0**;**//SERÁ REUTILIZADA, POR LO TANTO ES NECESARIO REINICIAR EL CONTADOR

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

**if** **(**posReinas**[**i**]==**j**){**//SE COLOCA UNA REINA

MoverCursor**(**x**,**y**);**

printf**(**"%c"**,**219**);**

**}else{**

**if((**j**+**aux**)%**2 **==** 0**){**//SE COLOCA UN "ESPACIO BLANCO"

MoverCursor**(**x**,**y**);**

printf**(**"%c"**,**176**);**

**}else{**

MoverCursor**(**x**,**y**);**

printf**(**"%c"**,**176**);**

**}**

**}**

x**+=**2**;**

**}**

y**+=**1**;**

aux**=**1**-**aux**;**

**\***alto **=** y**+**2**;**

**}**

printf**(**"\n"**);**

**}**

La función *mostrarTablero* imprime el tablero gracias al arreglo de posiciones, mediante una sentencia *for* se recorre dicho arreglo, se coloca el cursor en las coordenadas correspondientes (x,y o alto) y se imprime el caracter correspondiente. Posteriormente se hacen los respectivos incrementos a las variables (x,y).

/\*

FUNCIÓN: Reina(int fila, int posReinas[], int \*alto).

RECIBE: int fila (FILA DONDE SE COLOCARÁ A LA NUEVA REINA), int[] posReinas (TABLERO DE POSICIONES DE LAS REINAS)

,int\* alto (ORDENADA PARA SITUAR EL CURSOR).

DEVUELVE: LLAMA A LA FUNCIÓN NUEVAMENTE, SEGÚN SEA EL CASO.

DESCRIPCIÓN: COLOCA UNA REINA EN EL TABLERO. LLAMA A LA FUNCIÓN validarPosicion CADA VEZ QUE LO HACES

OBSERVACIONES:

\*/

void Reina **(**int fila**,** int posReinas**[],** int n**,** int **\***alto**)**

**{**

int ok **=** FALSE**;**

**if** **(**fila**<**n**)** **{**

// Quedan reinas por colocar

**for** **(**posReinas**[**fila**]=**0**;**posReinas**[**fila**]<**n**;**posReinas**[**fila**]++)** **{**

//COMPROBAMOS SI LA POSICIÓN ES VÁLIDA ¿PUEDO MANDAR A IMPRIMIR?

mostrarTablero**(**posReinas**,**n**,**alto**);**

EsperarMiliSeg**(**400**);**

**if** **(**validarPosicion**(**fila**,**posReinas**,**n**))** **{**

//SI LA POSICIÓN ES VÁLIDA, PROCEDEMOS A COLOCAR LA(S) SIGUIENTE(S) REINA(S)

Reina**(**fila**+**1**,**posReinas**,**n**,**alto**);**

**}**

//AQUÍ SE HACE EL BACKTRACKING, EN CASO DE NO SER VÁLIDA, SIMPLEMENTE NO SE LLAMA A LA FUNCIÓN Y SE SIGUE EL for

**}**

**}** **else** **{**

printf**(**"SOLUCION"**);**

**}**

**return;**

**}**

La función *Reina* es una función recursiva, que en caso de ser válida la posición para una reina agregada al arreglo se llama a sí misma para agregar una nueva reina al tablero. En caso de que el número de reinas agregadas sea igual al número de reinas por agregar y todas las posiciones sean válidas, se dice que se ha encontrado una solución al problema.

int main **(){**

int numReinas**;** // Número de reinas

int alto**=**0**;**

printf**(**"--------------------\nBIENVENIDO\n--------------------\n"**);**

printf**(**"Ingresa la cantidad de reinas a colocar (4<=numero<=10):\n>"**);**

scanf**(**"%i"**,** **&**numReinas**);**

int posReinas**[**numReinas**];**

system**(**"cls"**);**

//INICIALIZAMOS EL ARREGLO CON UN VALOR NEGATIVO (NO ENTRA EN EL RANGO DEL ARREGLO) PARA FACILITAR SU MANIPULACIÓN

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

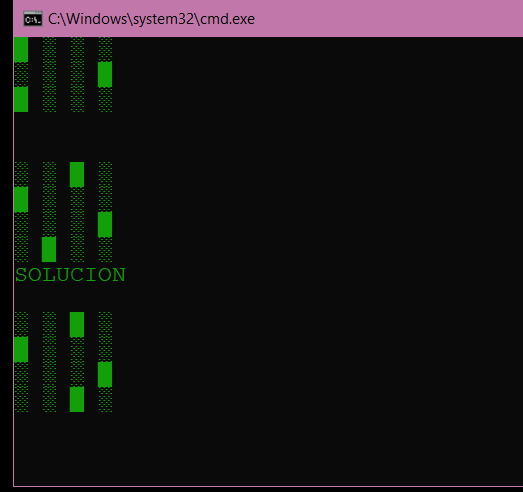
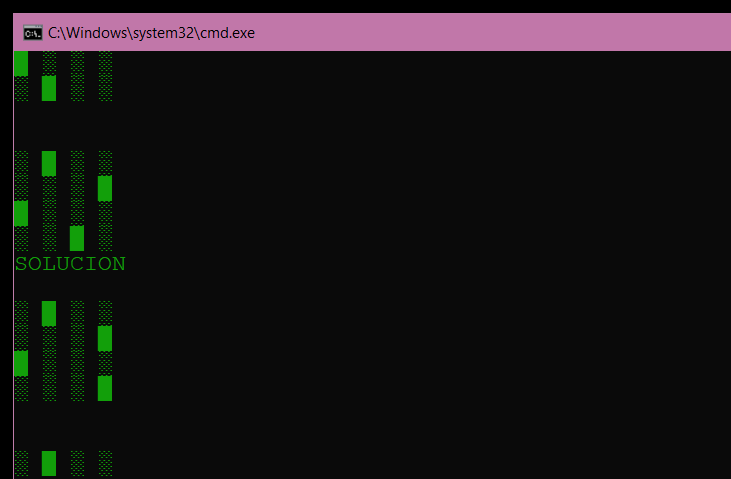
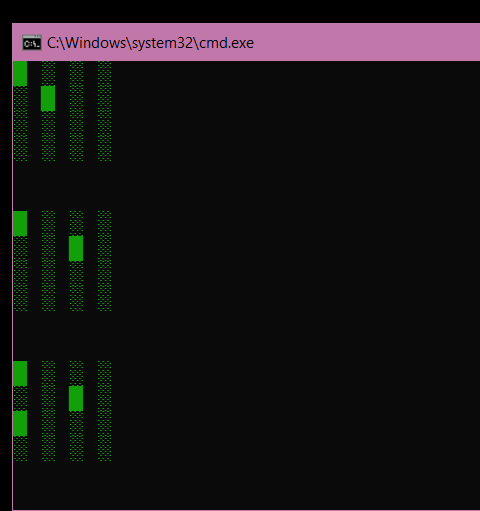
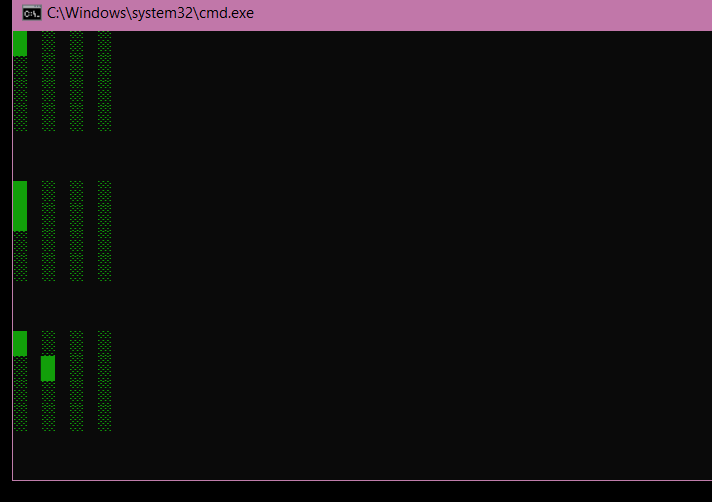
posReinas**[**i**]** **=** **-**1**;**

//COMENZAMOS CON LA RECURSIVIDAD PARA COLOCAR LAS REINAS

Reina**(**0**,**posReinas**,**numReinas**,&**alto**);**

**}**

**FUNCIONAMIENTO**



**ERRORES DETECTADOS**

Como observación se tiene que el programa puede limpiar la pantalla cada vez que imprima un nuevo tablero.

**POSIBLES MEJORAS**

El ingresar un número de reinas muy grande involucra un gran número de soluciones para el tablero, por lo tanto, el tiempo de ejecución se prolonga indefinidamente. Es posible modificar la manera en la que el programa imprime el camino para llegar a la solución, imprimiendo un solo tablero y limpiando la pantalla.

**CONCLUSIONES**

**Sánchez Castro Aarón Gamaliel:** Gracias a los conocimientos adquiridos de la práctica anterior es posible dar una solución recursiva para este problema utilizando la técnica *backtracking.* Una observación importante es que la eficiencia del algoritmo es directamente proporcional al tamaño del tablero especificado.