Problema de las 8 Reinas
Programación IV
Laboratorio: 1

Héctor F. Jiménez Saldarriaga
hfjimenez@utp.edu.co
PGP KEY ID: 0xB05AD7B8

Fecha de Entrega: **14** Febrero,2016
Profesor: Ing. Ángel Augusto Agudelo Zapata

1 Restricciones de Objetivo

Analizar, diseñar e implementar el problema de las 8 reinas en el lenguaje deseado.

- El tamaño del tablero debe ser variable.
- Debe tomar tiempos para saber lo que se demora el algoritmo seleccionado.
- Debe mostrar el resultado en pantalla
- Debe generar una animación

2 Análisis de Problema

Para entender el problema a realizar se hizo la lectura del articulo provisto por el docente a cargo del autor Niklaus Wirth[1] en el articulo se plantea en como se debería de llevar acabo un diseño cuidadoso y modular cuando se intenta resolver un problema, el problema en contexto es el análisis del problema de las 8 reinas utilizando *step refinement*, la idea es descomponer la solución planteada tanto como sea posible, estableciendo adecuadamente las p posibles condiciones que acortan la solución. Para el problema de las 8 reinas ***no existen*** soluciones analíticas y se deben explorar todas las posibles soluciones esto por obvias razones una solución por fuerza bruta seria costosa temporal y especialmente dado

1. https://www.inf.ethz.ch/personal/wirth/Articles/StepwiseRefinement.pdf

que tomaría $\binom{64}{8} = \frac{64!}{8!(64-8)!} = 4,426,165,368$ combinaciones, por tal motivo es necesario reducir la búsqueda de soluciones a una que cumpla los requisitos y condiciones establecidas por nosotros; utilizando *shortcuts* como lo menciona Niklaus tenemos en cuenta el movimiento ofensivo de la reina que puede atacar a i elementos en su fila y columna correspondiente, así que solo debe de haber una y solo una reina por fila,columna x, y del tablero, con esto podemos deducir que debemos permutar la n cantidad de reinas que deseemos poner en el tablero, reduciendo esto a un número inferior de posibles 40,320 posibles ubicaciones, esto corresponde a $8! = 40320$. Nuestra próxima condición será verificar los ataque en diagonales, que para para cada permutación hallada, que representa un conjunto de posibles soluciones que satisfacen que ninguna reina se ataque a nivel horizontal y vertical; si hay almenos una solución dejaremos de buscar más sobre el conjunto de permutaciones.

3 Solución del Problema

Teniendo en cuenta los hechos mencionados anteriormente asumiré que mi tablero máximo será un tablero de $n = 10$ ya que debo acotar la solución de mi problema, además encontrar la cantidad de permutaciones en c++ no parece una tarea tan fácil con un n mayor a 10, como lo sería con el paquete **itertools** de **Python3**. Para resolver este problema he decidido utilizar **c++11**, para medir el tiempo utilizare la librería chrono[2] y lo contrastare con el tiempo que me tire la utilidad **time** de Unix y presente en sistemas GnuLinux, para realizar la animación he pensado utilizar ncurses(**sin confirmar por que tengo unos problemas para el render:**) en su ultima versión, dado que la idea es ver como se realiza el posicionamiento de las reinas en el tablero y no algo muy gráfico como sería hacerlo con allegro.

4 Anexos

Los archivos correspondientes al proyecto de este laboratorio, junto con su readme y opciones soportadas pueden ser descargados del repositorio [3]

2. `http://www.cplusplus.com/reference/chrono/`
3. `https://github.com/h3ct0rjs/ProgrammingIVassignments/tree/master/Lab1`