Campus Santa Fe

Escuela de Ingeniería y Ciencias

Avenida: Computación y Tecnologías de la Información

TC1031.101

Programación de Estructuras de Datos y Algoritmos Fundamentales

Docente: Dr. Leonardo Chang Fernández

Actividad 2.3 – Actividad Integral estructura de datos lineales (Evidencia Competencia)

Axel Mercado Gasque

A00829051

Fecha de entrega: 13 de octubre del 2020

ADT de la clase (Lenguaje C++)

```
// Developed by: Axel Mercado Gasque
// Start Date: October 10th 2020
// End date: October 14th 2020

/* ********************
    This file contains the definition of the class "ConexionesComputadora"
   ******************** */

#ifndef ConexionesComputadora_hpp
#define ConexionesComputadora_hpp

#include <vector>
#include <string>
#include <stack> // Library for data structure stack is included (pila)
#include <queue> // Library for data structure queue is included (fila o
cola)
#include "Conexion.hpp"
using namespace std;


class ConexionesComputadora {

    private:
        string IP;
        string nombre;
        stack<Conexion> conexionesEntrantes;
        queue<Conexion> conexionesSalientes;

    public:
        // Constructor and destructor
        ConexionesComputadora();
        ~ConexionesComputadora();

        // Get methods
        string getIP();
        string getNombre();
        stack<Conexion> getConexionesEntrantes();
        queue<Conexion> getConexionesSalientes();
```

```
        // Set Methods
        void setIP(string);
        void setNombre(string);
        void setConexionesEntrantes(stack<Conexion>);
        void setConexionesSalientes(queue<Conexion>);


        // Additional Methods

        // Method that asks the user to enter either known IP or last
digits of IP
        void ingresaIP(vector<Conexion> &conexiones);

        // Method that reads file and stores IP addresses into vectors
        void leerArchivo(vector<Conexion> &conexiones);

        // Method that stores outgoing and incoming connections into
object
        void guardarConexiones(vector<Conexion> &conexiones);

        // Method that returns last incoming connection
        string ultimaConexionEntrante();

        // Method that returns first outgoing connection
        string primeraConexionSaliente();

        // Method that returns the number of incoming connections
        int numConexionesEntrantes();

        // Method that returns the number of outgoing connections
        int numConexionesSalientes();

        // Method that returns if an IP has three consecutive connections
to one web server
        string tresConexionesSeguidas();


};

#endif
```

Justificación de las estructuras de datos seleccionadas

Antes de justificar las estructuras de datos seleccionadas para modelar las conexiones entrantes y salientes de una computadora, es importante mencionar en que consiste esta etapa del reto. Esta etapa del reto consiste en implementar un ADT que modele las conexiones entrantes y salientes de una computadora. Para lograr esto, el programa debía primero generar o solicitar una dirección IP para posteriormente leer un archivo en formato csv, el cual contiene conexiones de varias computadoras. En este archivo se debían buscar las conexiones entrantes (en donde el IP generado o solicitado es el IP destino de la conexión) y las conexiones salientes (en donde el IP generado o solicitado es el IP origen de la conexión). El objetivo principal fue el implementar estructuras de datos que permitiesen acceder de manera eficiente a las conexiones entrantes de la última a la primera y acceder a las conexiones salientes de la primera a la última. Tomando en consideración lo mencionado anteriormente, se consideraron dos estructuras de datos lineales:

Stack (Pila) – Ya que se rige de la manera LIFO (Last In First Out).

Queue (Fila o Cola) – Ya que se rige de la manera FIFO (First in First Out)

Se seleccionó la estructura de datos lineal pila, ya que permite acceder de manera eficiente (en orden constante O(1)) al último elemento en haber sido insertado en la estructura. De esta manera, la pila es la estructura de datos lineal más adecuada para almacenar las conexiones entrantes, ya que se quiere acceder de la manera más eficiente posible a la última conexión entrantes que recibe una computadora. La razón por la cual se seleccionó esta estructura de datos es que es la que nos permite acceder al último elemento insertado en orden constante O(1). Si se usar alguna otra estructura de datos, quizá seguiría siendo posible acceder al último elemento insertado, pero el tiempo para hacer esto sería mayor.

De igual manera, se seleccionó la estructura de datos lineal fila o cola, ya que permite acceder de manera eficiente (en orden constante O(1)) al primer elemento en haber sido insertado en la estructura. De esta manera, la fila o cola es la estructura de datos lineal más adecuada para almacenar las conexiones salientes, ya que se quiere acceder de la manera más eficiente posible a la primera conexión saliente de una determinada computadora. La razón por la cual se seleccionó esta estructura de datos es que es la que nos permite acceder al último elemento insertado en orden constante O(1). Si se usar alguna otra estructura de datos, quizá seguiría siendo posible acceder al primer elemento insertado, pero el tiempo para hacer esto sería mayor.