INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE MONTERREY

AGO-DIC 2020

Actividad integral 2.3

Programación de estructuras de datos y algoritmos fundamentales (Gpo 101)

Integrantes del equipo:
Santiago Kohn Espinosa - A01029109
Pablo Yamamoto Magaña - A01022382

16 / Octubre / 2020

En la Imágen #1 se encuentra la nueva clase que desarrollamos para esta entrega. Para empezar utilizamos un stack de la librería STL para almacenar las conexiones entrantes y un queue de la misma librería para las conexiones salientes. Escogimos un stack ya que el problema requería leer las conexiones desde la última a la primera, esta descripción corresponde con la de una pila y la ideología detrás de este tipo de estructura de dato (FILO, First In, Last Out). El problema requería lo contrario para las conexiones salientes, por lo tanto utilizamos una cola (FIFO, First In, First Out).

```
template<class T>
#pragma once
class ConexionesComputadora
{
    string ip, nombre;
    stack<string> conexionesEntrantes;
    queue<string> conexionesSalientes;

public:
    ConexionesComputadora(){};
    ConexionesComputadora(string ip, string nombre, DataBase<T> *vector):ip{ip},nombre{nombre}
    {
        for (int i = 0; i < vector->All_Registrations.size(); i++)
        {
            if (vector->All_Registrations.at(i).getDestination_IP() == ip)
                conexionesEntrantes.push(vector->All_Registrations.at(i).getSource_IP());
            if (vector->All_Registrations.at(i).getSource_IP() == ip)
                conexionesSalientes.push(vector->All_Registrations.at(i).getDestination_IP());
        }
    }
    ~ConexionesComputadora(){}

    stack<string>* get_conexionesEntrantes(){return &conexionesEntrantes;}
    queue<string>* get_conexionesSalientes(){return &conexionesSalientes;}
```

Imagen 1. Clase ConexionesComputadora con los atributos ip, nombre, conexionesEntrantes y conexionesSalientes.

En cuanto a la complejidad temporal de estas dos estructuras empleadas, para acceder a los datos ambos tienen notación O(n), para insertar o borrar un dato tienen notación O(1), es constante en todos los casos.

En el constructor de ConexionesComputadora, mandamos el ip que elegí el usuario, el nombre, y los registros que leemos en la clase DataBase. Preferimos utilizar la lista de inicialización del constructor en lugar del cuerpo para darle valores a las variables ya que es mas eficiente de esta manera.

Para crear la ip asignada por el usuario creamos la instancia donde se contienen todo los registros e hicimos un do while para que no haya errores dentro del programa por error del input. Ya que tenemos el número ingresado por el usuario se lo sumamos a la ip de la empresa, obtenida en la actividad anterior.

```
string file_name = "equipo7.csv";
DataBase <Registration>DB(file_name, ',');
int num;
bool it = true;
do{
    cout << "Ingrese un numero entre el 1 y 150 (13 aparece en el documento): ";
    cin >> num;
    if (num > 0 && num < 151)
        it = false;
    else
        cout << "El numero escogido esta fuera del rango, intentelo de nuevo." << endl;
}while(it);

string ip_address = DB.company_ip()+to_string(num);
```

Después con la ip generada podemos obtener el nombre de la computadora haciendo una búsqueda secuencial, método utilizado porque las ip no las tenemos organizadas de ninguna manera, encontrando la ip obtenemos el nombre asignado en el registro.

```
int i = 0;
string name;
while (i != DB.All_Registrations.size())
{
    if (DB.All_Registrations.at(i).getSource_IP() == ip_address)
    {
        name = DB.All_Registrations.at(i).getSource_Hostname();
        break;
    }
    i++;
}
```

Creamos la instancia de “ConexionesComputadra”, mandando la ip, nombre y nuestra instancia donde tenemos todos los registros.

```
ConexionesComputadora <Registration> computadora(ip_address, name, &DB);
```

Imagen 2. Creación de la instancia computadora de tipo ConexionesComputadora.

```
string ultima = computadora.get_conexionesEntrantes()->top();
cout << "La última conexión que recibió fue de: " <<  ultima << endl;

size_t last_digit = ultima.find_last_of(".\\");
if (ultima.substr(0, last_digit) == ip_address.substr(0, last_digit))
    cout << "La conexión es Interna." << endl << endl;
else
    cout << "La conexión es Externa." << endl << endl;

cout << "Esta computadora tiene " << computadora.get_conexionesEntrantes()->size() << " conexiones entrantes." << endl;
cout << "Esta computadora tiene " << computadora.get_conexionesSalientes()->size() << " conexiones salientes." << endl;
```

Imagen 3. Obteniendo si la conexión es Interna o Externa

En la Imagen 3, guardamos en un string la última conexión que tiene nuestro stack, esto lo hacemos con la función top. Después obtenemos el índice en donde se encuentra el último punto de esta conexión del string, con este índice hacemos podemos utilizar la función substring para comparar si la última conexión es igual a la que generó el usuario. Si son iguales esto indica que la conexión es Interna, de otra forma es una conexión Externa.

Para la pregunta extra lo decidimos hacer dentro de una función para poder terminarlo en el caso de que sea true. Hicimos 2 implementaciones una para el caso de que sean exactamente 3 ip seguidas y otra donde le puedas mandar n y verifique si hay n conexiones seguidas.

Para la primera implementación sería O(n) iteramos por el queue una vez y mientras vamos avanzando borramos el valor al llegar al tercer valor igual, en el segundo if, regresamos el valor true.

La segunda implementación un poco más extra se podría considerar $O(n^2)$ al tener un while dentro de otro while. En esta implementación seguimos recorriendo el vector una sola vez iterando y teniendo un contador que nos dirá cuantas veces se repitió esa ip.

En cualquiera de los dos casos si se borra el queue la función regresa false.

```
bool preguntaExtra(int n)
{
    string ip;
    int i = 1;
    while(!conexionesSalientes.empty())
    {
        ip = conexionesSalientes.front();
        conexionesSalientes.pop();
        // Algoritmo que identifica exactamente 3 ip seguidas.
        /*
        if (!conexionesSalientes.empty() && ip == conexionesSalientes.front())
        {
            conexionesSalientes.pop();
            if (!conexionesSalientes.empty() && ip == conexionesSalientes.front())
                return true;
        }
        */
        while(i<n){
            if (!conexionesSalientes.empty() && ip == conexionesSalientes.front())
                conexionesSalientes.pop();
            else
            {
                i = 1;
                break;
            }
            i++;
        }
        if (i == n)
            return true;
    }
    return false;
}
```