Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Santa Fe

Act 2.3 - Actividad Integral estructura de datos lineales (Evidencia Competencia)

José Granger Salgado - A01023661
Jorge Cabiedes A01024053

16/10/2020

```cpp
#include "ADT_read.h"

using namespace std;


class ConexionesComputadora{
    public:
        string ip;
        string name = "";
        stack<Entry> conexionesEntrantes;
        queue<Entry> conexionesSalientes;
        vector<string> sitiosWeb;
        bool threeReps = false;


        void generateIP(string intIP)
        {
            int num;

            while(num > 150 || num < 1)
            {
                cout << "Porfavor ingrese un número entre 1 y 150: ";
                cin >> num;
                if (num > 150 || num < 1)
                    cout << "Ese no es un número valido porfavor intente de nuevo" << endl << endl;
            }

            ip = intIP + to_string(num);
        }

        void conexiones(vector<Entry> entries)
        {

            int n = 0;
            string temp = "";
            for (size_t i = 0; i < entries.size(); i++)
            {
                if (name == "" && (ip == entries[i].orIP || ip == entries[i].desIP))
                    name = entries[i].Name;

                if (entries[i].mail != "-" && (ip == entries[i].orIP || ip == entries[i].desIP))
                {
                    if(temp == "")
                        temp = entries[i].mail;
                    else if(entries[i].mail == temp)
                        n++;
                    else
                        n = 0;
                }

                if(ip == entries[i].orIP)
                    conexionesSalientes.push(entries[i]);
                else if(ip == entries[i].desIP)
                    conexionesEntrantes.push(entries[i]);

                if (n == 2)
                    threeReps = true;
            }

        }


};
```

Nuestra clase la llamamos Conexiones Computadora debido a las instrucciones establecidas, creamos 6 atributos en nuestra clase:

String ip es donde guardamos el string que creamos, donde tomamos la dirección interna creada.

String name es donde ponemos el nombre de la computadora que elegimos en este caso, la cual fue la primera que se encuentra dependiendo de nuestro ip, según las instrucciones de nuestro profesor.

stack<Entry> conexionesEntrantes es donde guardamos las conexiones entrantes, usamos un stack debido a que en las instrucciones de este atributo en específico, determinan que tenemos que poder agarrar los valores de ultimo a primero de una forma eficiente. stack (o Pilas) es un last-in-first out, por lo cual cumple perfectamente con nuestras especificaciones y lo hace de una forma eficiente en tiempo y espacio.

queue<Entry> conexionesSalidas es donde guardamos las conexiones salientes, usamos un queue (o Colas), debido a que en las instrucciones, necesitamos guardar y extraer de primera a última, por lo cual queue, como un first-in-first-out, es justo lo que necesitamos para cumplir con la optimización de espacio y tiempo.

bool threeReps helps us when we are doing the method check for the extra question.

```
string ip;
string name = "";
stack<Entry> conexionesEntrantes;
queue<Entry> conexionesSalientes;
bool threeReps = false;
```

For the first method, we generated an IP, where we let the user define the ending of our internal ip, with a range from 1 to 150, we use a while and if in order to determine whether the number the user enters is actually in the accepted range.

```
void generateIP(string intIP)
        {
            int num;

            while(num > 150 || num < 1)
            {
                cout << "Porfavor ingrese un número entre 1 y 150: ";
                cin >> num;
                if (num > 150 || num < 1)
                    cout << "Ese no es un número valido porfavor
intente de nuevo" << endl << endl;
            }

            ip = intIP + to_string(num);
        }
```

El último método que implementamos es conexiones, donde hacemos todo lo necesario para crear los atributos de nuestra clase y completar las preguntas.

Primero usamos un for para crear el loop donde se va a hacer todo nuestro código, después usamos un if para determinar el nombre dependiente de la primera vez que aparece el ip en nuestro csv.

Después usamos otro if para determinar la pregunta extra, donde tenemos que encontrar si se usan dos páginas en secuencia. En condiciones nos aseguramos que estamos evitando los valores vacíos.

Finalmente, creamos dos ifs donde comparamos el ip que creamos y los ips de origen y destino y los añadimos dependiendo de su lugar al stack o al queue, donde los podemos usar para completar el resto de las preguntas.

```
 void conexiones(vector<Entry> entries)
        {

            int n = 0;
            string temp = "";
            for (size_t i = 0; i < entries.size(); i++)
            {
                if (name == "" && (ip == entries[i].orIP || ip ==
entries[i].desIP))
                    name = entries[i].Name;

                if (entries[i].mail != "-" && (ip == entries[i].orIP ||
ip == entries[i].desIP))
                {
                    if(temp == "")
                        temp = entries[i].mail;
                    else if(entries[i].mail == temp)
                        n++;
                    else
                        n = 0;
                }

                if(ip == entries[i].orIP)
                    conexionesSalientes.push(entries[i]);
                else if(ip == entries[i].desIP)
                    conexionesEntrantes.push(entries[i]);

                if (n == 2)
                    threeReps = true;
            }

        }
```

Time complexities and the result of the questions can be found on the code.