ACT 5.2

Elías Uriel Velázquez Rojas

A01639716 IRS

El inicio del programa creamos una clase llamada Hashtable, a su vez tambien se crea un enum y una estructura, el enum tiene valor, ocupado, y borrado, y la estructura que tiene los atributos de accesos, ip info y el enum estado que se declara como vacío.

En la clase los atributos privados, son la lista celda que se llama table, y otra variable entera llamado total_elements y en publico se declara el constructor y la función para insertar los elementos, que usa los accesos de la ip y con el push back se guarda,

```
#pragma once
#include <iostream>
#include <list>
using namespace std;

enum estado {
    vacio, ocupado, borrado
};

struct celda {
    int accesos;
    string ip;
    string info;
    estado miEstado = vacio;
};

class HashTable
{
private:
    list<celda>* table;
    int total_elements;

public:
    HashTable(int n) {
        total_elements = n;
        table = new list<celda>[total_elements];
    }

    void insertElement(celda ip) {
        table[ip.accesos].push_back(ip);
    }
```

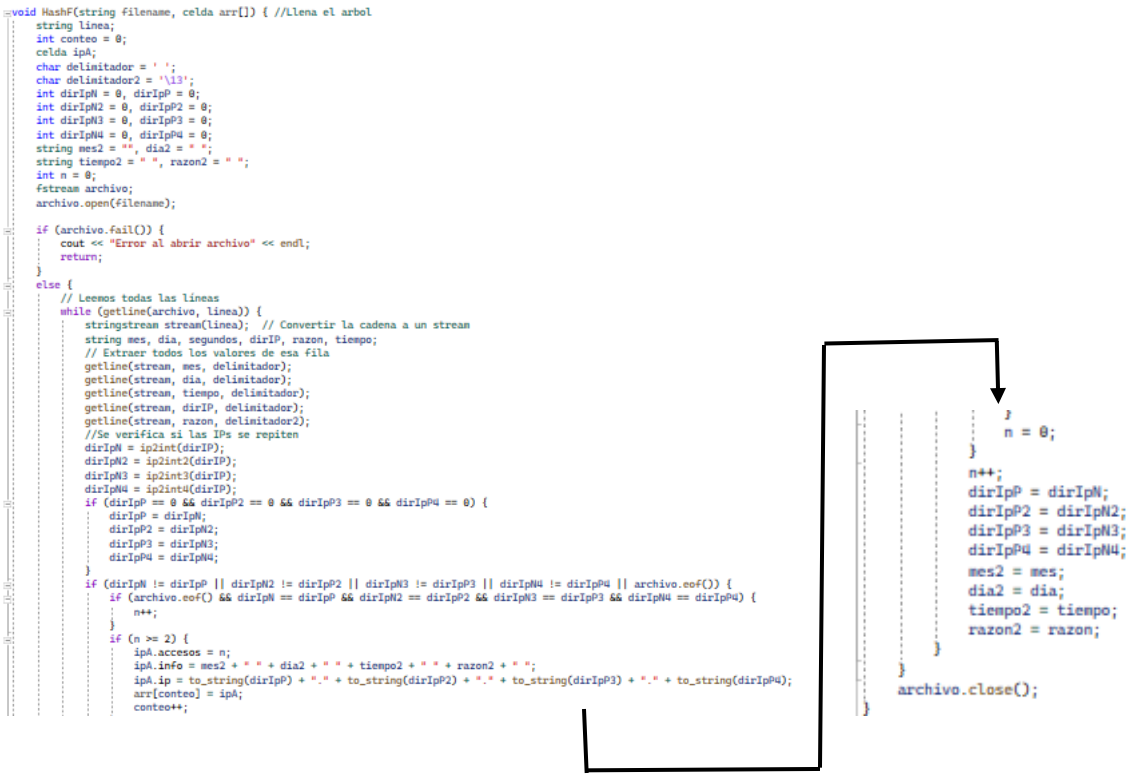
La funcion buscarIP, sirve como su nombre lo dice para buscar las ip, con el uso de dos for uno que recorra los elementos y otro para el índice, que lo vaya recorriendo y se usa un contador para hacer una verificación para ver si se ha encontrado un a ip, después se usa otra función para imprimir todo igual usando dos for con el mismo uso.

```
    void buscarIP(string ip) { // O (n^2)
        int p = 0;
        for (int i = 0; i < total_elements; i++) {
            for (celda j : table[i]) {
                if (j.ip == ip) {
                    cout << "Index " << i << ": ";
                    cout << j.ip << " " << j.info;
                    cout << endl;
                    p++;
                }
            }
        }
        if (p == 0)
            cout << "IP no encontrada" << endl;
    }

    void printAll() { // O (n^2)
        for (int i = 0; i < total_elements; i++) {
            cout << "Index " << i << ": ";
            for (celda j : table[i])
                cout << j.ip << " -> ";

            cout << endl;
        }
    }
};
```

Se incluyen las librerías, después se crea las funciones para regresar los valores enteros por ip, esto mediante el uso de la librería sstream, se declara variables que después se asigna a una variable rst y se retorna, eso se hace con los 4 valores de la ip.

```
#include <iostream>
using namespace std;
#include <vector>
#include <fstream>
#include <iostream>
#include <cstdlib>
#include <string>
#include <bits.h>
#include <sstream>
#include <chrono>
#include <thread>
using namespace std::this_thread;
#include "HashTable.h"

int ip2int(string ip) { //Regresa primer valor entero de IP
    int rst = 0;
    int a, b, c, d;
    char t = '.';
    stringstream ss(ip);
    ss >> a >> t >> b >> t >> c >> t >> d;
    rst = (a);
    return rst;
}

int ip2int2(string ip) { //Regresa segundo valor entero de IP
    int rst = 0;
    int a, b, c, d;
    char t = '.';
    stringstream ss(ip);
    ss >> a >> t >> b >> t >> c >> t >> d;
    rst = (b);
    return rst;
}

int ip2int3(string ip) { //Regresa tercer valor entero de IP
    int rst = 0;
    int a, b, c, d;
    char t = '.';
    stringstream ss(ip);
    ss >> a >> t >> b >> t >> c >> t >> d;
    rst = (c);
    return rst;
}

int ip2int4(string ip) { //Regresa cuarto valor entero de IP
    int rst = 0;
    int a, b, c, d;
    char t = '.';
    stringstream ss(ip);
```

Esta función se usa para rellenar de información proveniente del txt dentro de la funcion se definen los delimitadores de tipo char, y las variables para guardar las direcciones de tipo entero, a su vez de los contadores, y con la función getline y después se verifica si las Ip se repiten luego se verifica las ip mediante la dirección de ip , si se verifica la dirección se va sumando el contador y la variable n, , delimitando la dirección de la ip, y las repeticiones con el valor de n, que se va sumando dependiendo la verificación.



```
void HashF(string filename, celda arr[]) { //Llena el arbol
    string linea;
    int conteo = 0;
    celda ipA;
    char delimitador = ' ';
    char delimitador2 = '\13';
    int dirIpN = 0, dirIpP = 0;
    int dirIpN2 = 0, dirIpP2 = 0;
    int dirIpN3 = 0, dirIpP3 = 0;
    int dirIpN4 = 0, dirIpP4 = 0;
    string mes2 = "", dia2 = " ";
    string tiempo2 = " ", razon2 = " ";
    int n = 0;
    fstream archivo;
    archivo.open(filename);

    if (archivo.fail()) {
        cout << "Error al abrir archivo" << endl;
        return;
    }
    else {
        // Leemos todas las líneas
        while (getline(archivo, linea)) {
            stringstream stream(linea);  // Convertir la cadena a un stream
            string mes, dia, segundos, dirIP, razon, tiempo;
            // Extraer todos los valores de esa fila
            getline(stream, mes, delimitador);
            getline(stream, dia, delimitador);
            getline(stream, tiempo, delimitador);
            getline(stream, dirIP, delimitador);
            getline(stream, razon, delimitador2);
            //Se verifica si las IPs se repiten
            dirIpN = ip2int(dirIP);
            dirIpN2 = ip2int2(dirIP);
            dirIpN3 = ip2int3(dirIP);
            dirIpN4 = ip2int4(dirIP);
            if (dirIpP == 0 && dirIpP2 == 0 && dirIpP3 == 0 && dirIpP4 == 0) {
                dirIpP = dirIpN;
                dirIpP2 = dirIpN2;
                dirIpP3 = dirIpN3;
                dirIpP4 = dirIpN4;
            }
            if (dirIpN != dirIpP || dirIpN2 != dirIpP2 || dirIpN3 != dirIpP3 || dirIpN4 != dirIpP4 || archivo.eof()) {
                if (archivo.eof() && dirIpN == dirIpP && dirIpN2 == dirIpP2 && dirIpN3 == dirIpP3 && dirIpN4 == dirIpP4) {
                    n++;
                }
                if (n >= 2) {
                    ipA.accesos = n;
                    ipA.info = mes2 + " " + dia2 + " " + tiempo2 + " " + razon2 + " ";
                    ipA.ip = to_string(dirIpP) + "." + to_string(dirIpP2) + "." + to_string(dirIpP3) + "." + to_string(dirIpP4);
                    arr[conteo] = ipA;
                    conteo++;
                    n = 0;
                }
                n++;
                dirIpP = dirIpN;
                dirIpP2 = dirIpN2;
                dirIpP3 = dirIpN3;
                dirIpP4 = dirIpN4;
                mes2 = mes;
                dia2 = dia;
                tiempo2 = tiempo;
                razon2 = razon;
            }
        }
        archivo.close();
    }
```

La función insert, que recibe un atributo de la estructura celda, después un arreglo del mismo otra variable entera recibe los acceso y lo asigna a índice y con un ciclo usando while va verificando si el estado del arreglo de acuerdo al índice esta vacio o borrado si es asi la variable bool se declara como true y se le asigna los valores al arreglo de acuerdo a la posición dicha por el índice y el estado en esa posición cambia a ocupado, sino se cumple la verificación el índice se declara, usando el mod de 10 sumándole uno anteriormente .

La función mostrar su única función es mostrar los valores que se encuentran dentro de la del arreglo de estructura celda que lo recibe y la cantidad de valores que hay dentro de el esto lo hace mediante un for.

```
void insert(celda ip, int cant, celda arr[]) {

    int indice = ip.accesos;
    bool ocupado_bool = false;

    while (ocupado_bool == false) {
        if (arr[indice].miEstado == vacio || arr[indice].miEstado == borrado) {
            ocupado_bool = true;
            arr[indice].ip = ip.ip;
            arr[indice].info = ip.info;
            arr[indice].miEstado = ocupado;
        }
        else {
            indice = (indice + 1)%10;
        }
    }
}

void mostrar(int cant, celda arr[]) {
    for (int n = 0; n < cant; n++) { // O (n)
        cout << n << " " << arr[n].ip << endl;
    }
}
```

Esta función se usa para encontrar la llave, que recibe tres variables como string, entero y el arreglo de celda, esto se logra mediante un for, y dentro de él se usa un if, se imprime las ip y la información del arreglo, y aumenta la variable p, si p es igual a cero no se encontró la ip.

```
void findKey(string ip, int cant, celda arr[]) {
    int p = 0;
    for (int n = 0; n < cant; n++) { // O (n)
        if (arr[n].ip == ip) {
            cout << "Index " << n << ": " << arr[n].ip << " " << arr[n].info << endl;
            p++;
        }
    }
    if (p == 0)
        cout << "IP no encontrada" << endl;
}
```

Se declaran las variables para crear el arreglo, la cantidad se abre el txt y los arreglos de celdas, a su vez que una variable opción para elegir que deseas hacer si es igual a 0, te dan otras dos opciones 1 para insertar valores y 4 para salir, sino te dan otros 4 opciones, igual insertar elementos luego ver hash tables, 3 buscar ip en las tablas y 4 para salir, luego hace mas verificaciones si es igual a 1 y listo es igual a 1, ya no podrías insertar más elementos, y sino te permite ingresar más elementos cambia los estados, ip, la información y los accesos.

```
int main()
{
    int cantidad = 10;
    celda arr[10];
    string filename;
    filename = "bitacora Bueno.txt";
    HashTable ht(10);
    celda array[10];
    int listo = 0;
    string IP ;
    int opcion = 0;
    int c;

    while (opcion != 4) {
        cout << "Actividad Integral sobre el uso de codigos hash" << endl;
        if (listo == 0) {
            cout << "*Para ver mas opciones, primero ingresa valores o datos a las tablas*" << endl;
            cout << endl;
            cout << "1. Insertar elementos de archivo en tablas de Hash" << endl;
            cout << "4. Salir" << endl;
        }
        else
        {
            cout << endl;
            cout << "1. Insertar elementos de archivo en tablas de Hash" << endl;
            cout << "2. Ver Hash Tables" << endl;
            cout << "3. Buscar IP en tablas" << endl;
            cout << "4. Salir" << endl;
        }
        cout << endl;
        cout << "Escribe la opcion deseada: ";
        cin >> opcion;
        if (opcion == 1) {
            if (listo == 1) {
                cout << endl;
                cout << "Ya no puedes insertar mas elementos a las tablas" << endl;
                cout << endl;
            }
            else {
                cout << endl;
                cout << "--- Insertar elementos del archivo en tablas de Hash ---" << endl;
                cout << endl;
                cout << "Agregando elementos en tabla ..." << endl;
                for (int i = 0; i < cantidad; i++) { // O (n)
                    arr[i].miEstado = vacio;
                    arr[i].ip = "";
                    arr[i].info = "";
                    arr[i].accesos = 0;
                }
```

Después se llama la función HashF y después se hice un for para insertar los valores recibidos, para después hacer lo mismo con los hash tables, con el arreglo, si se crearon correctamente imprime la pantalla, y se aumenta el contador de listo y se termina esa verificación, después si se escoge la opción dos dentro de esta se manda a llamar la función de hashtable para imprimir que es Hash Table Chain y después con la función mostrar la Hash Table Prueba Lineal, si la opción es igual a 3 te pide ingresar los datos ingresados correctamente y te pide la IP, después buscar esa ip dentro de la Hash Table Chain, al igual de encontrar la llave en la prueba lineal, y para terminar el 4 para salir, sino se ingresa la cantidad correcta te pide volverla a escribir.

```
        HashF(filename, arr);

        for (int i = 0; i < cantidad; i++) { // O (n)
            insert(arr[i], cantidad, array);
        }

        for (int i = 0; i < 10; i++) // O (n)
            ht.insertElement(arr[i]);

        sleep_for(1s);
        cout << endl;
        cout << "Hash tables generados correctamente" << endl;
        listo++;
        cout << endl;
    }
}
else if (opcion == 2) {
    cout << endl;
    cout << "-*- Hash Tables -*-" << endl;
    cout << endl;
    cout << "<< Hash Table Chain >>" << endl;
    cout << endl;
    ht.printAll();
    cout << endl;
    cout << "<< Hash Table Prueba lineal >>" << endl;
    cout << endl;
    mostrar(cantidad, array);
    cout << endl;
}
else if (opcion == 3) {
    cout << endl;
    cout << "Introduce correctamente la IP a buscar (sin el puerto, sin :xxxx): " << endl;
    cout << "SI -> 1.6.378.65, NO -> 1.6.378.65:6772" << endl;
    cout << ">>> ";
    cin >> IP;
    cout << endl;
    cout << "<< Hash Table Chain >>" << endl;
    cout << endl;
    ht.buscarIP(IP);
    cout << endl;
    cout << "<< Hash Table Prueba lineal >>" << endl;
    cout << endl;
    findKey(IP, cantidad, array);
    cout << endl;
```

```
        else if (opcion == 4) {
            cout << endl;
            cout << "Saliendo..." << endl;
            break;
        }
        else{
            cout << "\nEnter correct option\n";
            cout << endl;
        }
    }
    return 0;
}
```

CASOS PRUEBAS:

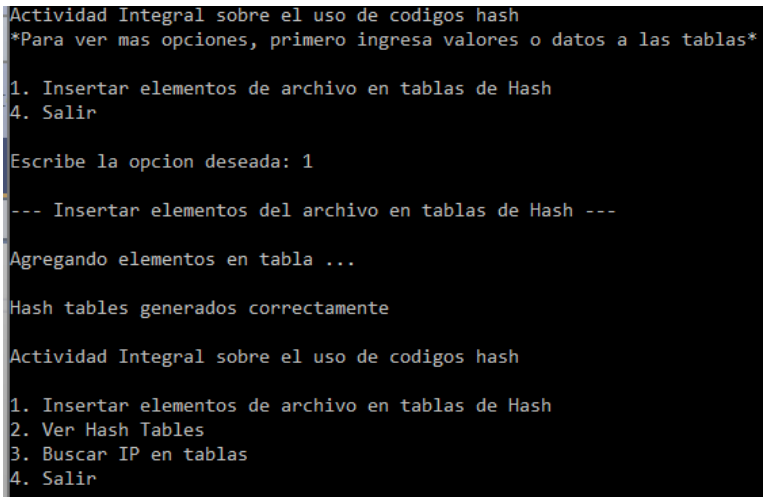
```
1. Insertar elementos de archivo en tablas de Hash
4. Salir

Escribe la opcion deseada: 1

--- Insertar elementos del archivo en tablas de Hash ---

Agregando elementos en tabla ...

Hash tables generados correctamente

Actividad Integral sobre el uso de codigos hash

1. Insertar elementos de archivo en tablas de Hash
2. Ver Hash Tables
3. Buscar IP en tablas
4. Salir

Escribe la opcion deseada: 2
```

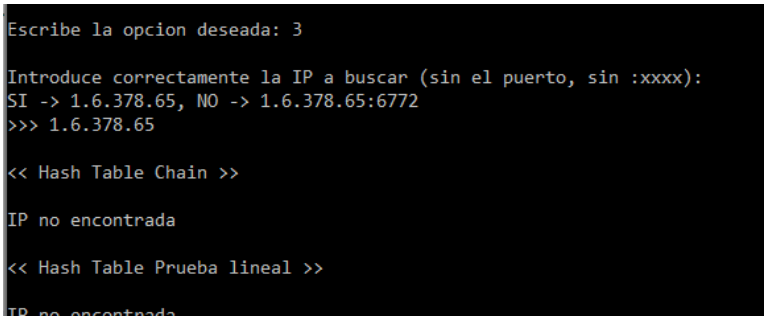
```
-*- Hash Tables -*-

<< Hash Table Chain >>

Index 0:
Index 1:
Index 2: 587.2.198.2 -> 434.81.982.62 ->
Index 3:
Index 4: 2.30.316.8 ->
Index 5: 34.60.781.17 -> 644.82.541.35 -> 155.37.606.21 ->
Index 6: 998.16.929.91 ->
Index 7: 253.57.262.14 ->
Index 8: 510.9.976.72 ->
Index 9: 787.75.330.45 ->

<< Hash Table Prueba lineal >>

0 787.75.330.45
1 155.37.606.21
2 587.2.198.2
3 434.81.982.62
4 2.30.316.8
5 34.60.781.17
6 644.82.541.35
7 253.57.262.14
8 510.9.976.72
9 998.16.929.91
```

Caso prueba 2:

```
Actividad Integral sobre el uso de codigos hash
*Para ver mas opciones, primero ingresa valores o datos a las tablas*

1. Insertar elementos de archivo en tablas de Hash
4. Salir

Escribe la opcion deseada: 1

--- Insertar elementos del archivo en tablas de Hash ---

Agregando elementos en tabla ...

Hash tables generados correctamente

Actividad Integral sobre el uso de codigos hash

1. Insertar elementos de archivo en tablas de Hash
2. Ver Hash Tables
3. Buscar IP en tablas
4. Salir

Escribe la opcion deseada: 1

Ya no puedes insertar mas elementos a las tablas
```

CASOS PRUEBA 3:





CASO PRUEBA 4: