

Este código en lenguaje C++ se encarga de leer las instancias originales de Karapetyan [25] y además crea un nuevo archivo .txt que contiene toda la información de la instancia original más los dos nuevos vectores y la cantidad de mercados, creados a través de ciclos y de condicionales apropiados. Estos vectores, son la demanda que está distribuida uniformemente entre 1 y 3, y la vulnerabilidad que se distribuye uniformemente también entre 1 y 5. El número de mercados es un aleatorio entre la suma de la demanda mínima de cada clúster y la suma de la demanda de todos los nodos. A medida que se avanza en el código se encuentran algunas descripciones de las funciones implementadas.

```

1  #include <iostream> // Librerías de C++.
2  #include <fstream>
3  #include <string>
4  #include<vector>
5  #include<locale>
6  #include<sstream>
7  #include <random>
8  #include <numeric>
9
10 using namespace std;
11 random_device rd;
12 mt19937 generator(rd());
13
14 vector<int> quitarChar(string s) // Recibe un string con valores y espacios.
    Devuelve un vector únicamente con los valores.
15 {
16     locale loc;
17     for (int i = 0; i < s.size(); i++) {
18         if (isdigit(s[i], loc) == false) { // Quita los espacios.
19             s[i] = ' ';
20         }
21     }
22     vector<int> v;
23     int aux;
24     stringstream daniel(s);
25     while (daniel >> aux) {
26         v.push_back(aux); // Los valores que estaban en el string van siendo
            puestos en el vector.
27     }
28     return v; // Retorna el vector.
29 }
30 int leer(int& N, int& M, vector<vector<int>>& mCluster, vector<vector<int>>&
    mDistance, string name) // Lee y obtiene la información del .txt.
31 {
32     fstream f;
33     string dire = "C:\\Users\\User\\Escritorio\\InstancesPD\\Originales\\" + name
        + ".txt"; // Obtiene los nombres de cada instancia que están en un .txt.
34     f.open(dire, fstream::in); // Abre el .txt con el nombre de la instancia.
35     if (f.is_open() == false) {
36         cout << "No se puede abrir el archivo de lectura.\n";
37         cout << dire << endl;
38         return -1; // Retorna un "No se puede abrir el archivo de lectura" en caso de no
            abrirse el archivo ".txt.
39     }

```

## ANEXO 4. CrearDatos

```
40
41     string aux;
42
43     f >> aux;
44     f >> N; // Lee el número de nodos de la instancia .txt.
45     f >> aux;
46     f >> M; // Lee el número de clústeres de la instancia .txt.
```



C:\Users\User\Escritorio\C++2\P1\P1\Header.h

3

```

88     }
89     }
90 void crearDatos(string nombre)
91 {
92     int N; // Número de nodos.
93     int M; // Número de clústeres.
94     int minDem = 1; // Parámetro mínimo de demanda.
95     int maxDem = 3; // Parámetro máximo de demanda.
96     int minVul = 1; // Parámetro mínimo de vulnerabilidad.
97     int maxVul = 5; // Parámetro máximo de vulnerabilidad.
98
99     string nombre2 = nombre;
100     nombre.pop_back(); nombre.pop_back(); nombre.pop_back(); nombre.pop_back();
101
102     88         ofstream k("C:\\Users\\User\\Escritorio\\InstancesPD\\Nuevas\\"
+nombre + "_COVID.txt"); // Crea el ""_COVID.txt donde van los datos leídos del
.txt de entrada, junto con los nuevos vectores de demanda y vulnerabilidad
correspondientes a cada nodo.
103
104     vector<vector<int>> mCluster; // Vector de vectores para los clústeres.
105     vector<vector<int>> mDistance; // Vector de vectores para las distancias
entre nodos.
106     vector<int> vDemanda; // Vector de demanda.
107     vector<int> vVulne; // Vector de vulnerabilidad.
108
109     leer(N, M, mCluster, mDistance, nombre); // Llama la función leer con las
referencias necesarias (N, M, mCluster, MDistancia, nombre).
110     k << N << endl; // Se escribe el número de nodos (N) en el nuevo .txt.
111     k << M << endl; // Se escribe el número de Clusters (M) en el nuevo .txt.
112
113     for (int i = 0; i < M + N; i++) {
114         if (i < M) {
115             mostrar(mCluster[i], k); // Llama la acción "mostrar" para que en el
nuevo .txt se escriban cada uno de los vectores del vector de
vectores m.Cluster.
116             k << endl;
117         }
118         else {
119             mostrar(mDistance[i - M], k); // Llama la acción "mostrar" para
que en el nuevo .txt se escriban cada uno de los vectores del
vector de vectores m.Distance.
120             k << endl;
121         }
122     }
123     int num = 4; /// DEMANDA DEL PRIMER NODO ALMACEN
124     alea(minDem, maxDem, N, vDemanda); // Llama la acción "alea" para crear el
nuevo vector de demanda, con un valor correspondiente para cado nodo.
125     alea(minVul, maxVul, N, vVulne); // Llama la acción "alea" para crear el nuevo
vector de vulnerabilidad, con un valor correspondiente para cado nodo.
126
127     vVulne.erase(vVulne.begin() + 0);

```

C:\Users\User\Escritorio\C++2\P1\P1\Header.h

4

```

128     vDemanda.erase(vDemanda.begin() + 0);
129     vVulne.insert(vVulne.begin(), 0);
130     vDemanda.insert(vDemanda.begin(), num);
131
132
133     mostrar(vDemanda, k); // Llama la acción "mostrar" para que en el
                            nuevo .txt se escriba el nuevo vector de demanda.
134     k << endl;
135
136     mostrar(vVulne, k); // Llama la acción "mostrar" para que en el nuevo .txt se
                            escriba el nuevo vector de vulnerabilidad.
137
138     vector <vector <int>> mClusDem;
139     vector <int> p;
140     vector <int> vMinDem;
141
142     for (int i = 0; i < mCluster.size(); i++) {
143         p.clear();
144         ; for (int j = 0; j < mCluster[i].size(); j++) {
145             p.push_back(vDemanda[mCluster[i][j] - 1]);
146         }
147         mClusDem.push_back(p);
148     }
149
150     double sumDemmax = accumulate(vDemanda.begin(), vDemanda.end(), 0) - num;
151     int b = 0;
152
153     for (int i = 0; i < mClusDem.size(); i++) {
154         int a = *min_element(mClusDem[i].begin(), mClusDem[i].end());
155         b += a;
156         vMinDem.push_back(a);
157     }
158
159     uniform_int_distribution<int> distrib(b, sumDemmax);
160     int nmerca = (distrib(generator));
161     int nmerca_inicial = nmerca;
162
163     k << endl;
164     k << nmerca << endl;
165
166     }
167
168     #include "Header.h"
169
170     int main() {
171         vector<string> vName; // Vector de strings que será lleno con los nombres
                                de las instancias
172         string aux;
173         fstream f("C:\\Users\\User\\Escritorio\\InstancesPD\\nombres.txt", fstream::in);
                                // Se lee el .txt con los nombres de cada una de las instancias
174

```

## ANEXO 4. CrearDatos

C:\Users\User\Escritorio\C++2\P1\P1\Header.h

5

```
175     while (f >> aux) {  
176         vName.push_back(aux); // Lee todos los nombres de las instancias y se  
                                // van poniendo en el vector de strings vName.  
177     }  
178     for (int i = 0; i < vName.size(); i++)  
179     {  
180         crearDatos(vName[i]); // Llama la acción "crearDatos" para cada uno de  
                                // los nombres presentes en el vector de strings  
181     }  
182  
183     cout << "fin." << endl; // Muestra "fin." en pantalla cuando haya terminado de  
                                // ejecutarse el programa.  
184     cin.get();  
185     return 0;  
186 }  
187
```