

LABORATORIO 5: M-TREE

Estudiante: García Cáceres Uberto

Grupo: B

CUI: 20200721

1. Implemente un M-Tree considerando los procedimientos de inserción y búsqueda.

```
1 #define M_TREE_H
2 #ifndef M_TREE_H
3 #include "m_tree_nodo.h"
4 > class m_tree{...
16 > m_tree::m_tree(int hijos_maximo = 4){...
20 > void m_tree::anadir(punto*nuevo){...
23 > ostream& operator << (ostream &o,const m_tree &p)...
28 > void m_tree::vecinos_mas_cercanos(punto*buscado, float distancia = 0, int cantidad = 1){...
31 > m_tree::~~m_tree()...
34 #endif
```

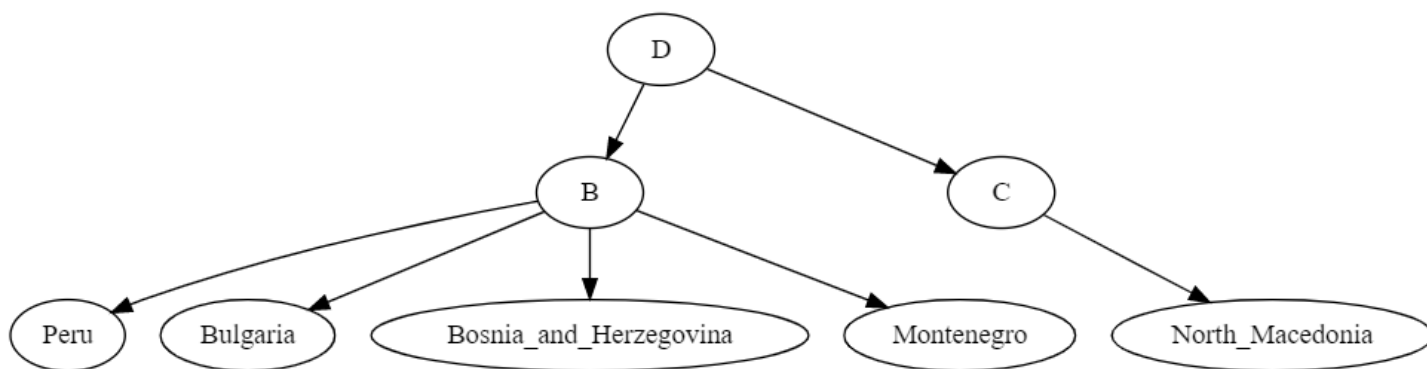
Lín. 34, col. 7 Espacios: 4 UTF-8 CRLF C++ Win32

```
1 #define M_TREE_NODO_H
2 #ifndef M_TREE_NODO_H
3 #include <iostream>
4 #include <limits.h>
5 #include <algorithm>
6 #include <vector>
7 #include <queue>
8 using namespace std;
9 > struct punto{...
14 > void llenar(punto*media, string n_1, int x_1, int y_1){...
19 > bool comparar(const pair<float,punto> &a, const pair<float,punto> &b){...
22 > class m_tree_nodo...
43 > m_tree_nodo::m_tree_nodo()...
50 > float m_tree_nodo::distancia(punto* a, punto* b){...
55 > float m_tree_nodo::distancia(vector<punto*> a, vector<punto*> b){...
90 > void m_tree_nodo::anadir(punto* nuevo_punto, int n_maximo){...
112 > void m_tree_nodo::actualizar(m_tree_nodo*nuevo, m_tree_nodo*hijo){...
185 > void m_tree_nodo::division(m_tree_nodo*nuevo, int n_maximo){...
234 > void m_tree_nodo::particion(m_tree_nodo*dividido,m_tree_nodo*part1,m_tree_nodo*part2){...
316 > ostream& operator << (ostream &o,const m_tree_nodo &p)...
334 > void m_tree_nodo::vecinos_mas_cercanos(punto* buscado, float distancia, int cantidad){...
396 m_tree_nodo::~~m_tree_nodo()
397 {
398 }
399 #endif
```

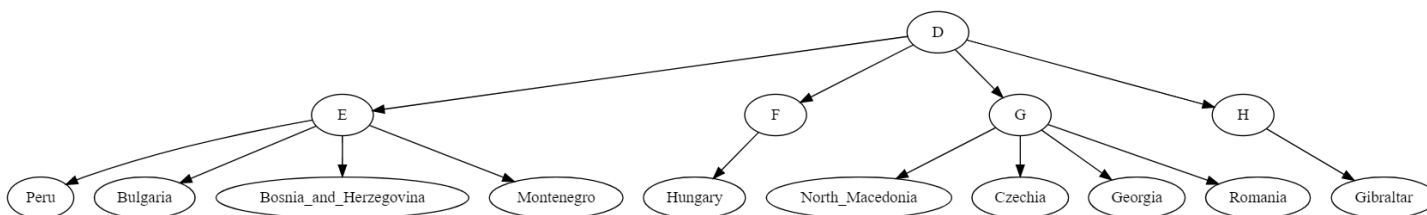
Lín. 4, col. 20 Espacios: 4 UTF-8 CRLF C++ Win32

2. Inserta conjunto de puntos proporcionado en la tabla 1 en el M-Tree y mostrar gráficamente el estado del árbol después de insertar:

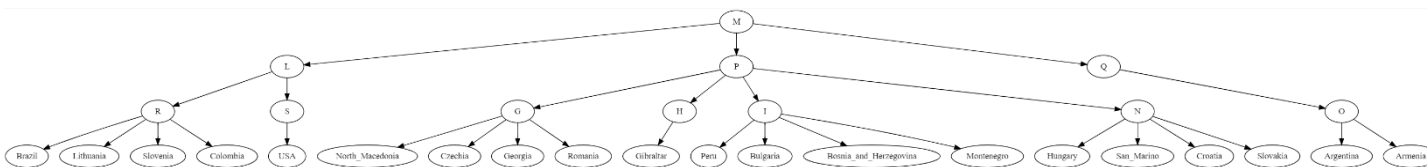
a) El quinto elemento de la tabla (North Macedonia)



b) El décimo elemento de la tabla (Gibraltar)



c) El último elemento de la tabla (USA)



3. Dada una distancia de 37000, ¿cuántos y cuáles son los países más próximos a Perú? (mostrar la búsqueda ejecutada y el resultado en el lenguaje de programación escogido)

No hay vecino cercano en dicho rango.

```

1  ,149455,124480,116439,113938,176235,202419,98156,148104};
2  float llenar_y[20] = {5983,4139,3870,3673,3639,3586,3080,3030,2966,2910,2863,2733,2678,2639,2547,2547
3  ,2525,2512,2489,2406};
4  for (int i = 0; i < 20; i++){
5      punto *a = new punto();
6      llenar(a,llenar_nombre[i],llenar_x[i],llenar_y[i]);
7      arbol_m.anadir(a);
8      delete a;
9      if(i == 4 || i == 9){
10         cout<<arbol_m;
11         cout<<" ";<<endl;
12     }
13 }
14 cout<<arbol_m;
15 cout<<" ";<<endl;
16 punto *a = new punto();
17 llenar(a,llenar_nombre[0],llenar_x[0],llenar_y[0]);
18 arbol_m.vecinos_mas_cercanos(a,37000);
19 cout<<" ";<<endl;
20 llenar(a,llenar_nombre[5],llenar_x[5],llenar_y[5]);
21 arbol_m.vecinos_mas_cercanos(a);
22
23 /*
24 cout<<" ";<<endl;
25 m_tree arbol_n;
26 string llenar_nombre2[20] = {"USA","Slovenia","Slovakia","San_Marino","Romania","Peru", "North_Macedonia","Montenegro",
27 "Lithuania","Hungary","Gibraltar","Georgia","Czechia","Croatia","Colombia","Bulgaria","Brazil","Bosnia and Herzegovina"};
28 */
  
```

```
9      ,149455,124480,116439,113938,176235,202419,98156,148104};
10     float llenar_y[20] = {5983,4139,3870,3673,3639,3586,3080,3030,2966,2910,2863,2733,2678,2639,2547,2547
11     ,2525,2512,2489,2406};
12     for (int i = 0; i < 20; i++){
13         punto *a = new punto();
14         llenar(a,llenar_nombre[i],llenar_x[i],llenar_y[i]);
15         arbol_m.anadir(a);
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL JUPYTER

G -> Czechia;
G -> Georgia;
G -> Romania;
H -> Gibraltar;
I -> Peru;
I -> Bulgaria;
I -> Bosnia_and_Herzegovina;
I -> Montenegro;
N -> Hungary;
N -> San_Marino;
N -> Croatia;
N -> Slovakia;
Q -> 0;
O -> Argentina;
O -> Armenia;

No hay vecino cercano en dicho rango

4. ¿Cuál es el país más próximo de Hungary? (mostrar la búsqueda ejecutada y el resultado en el lenguaje de programación escogido)

Armenia

```
22     cout<<arbol_m;
23     cout<<"
24     punto *a = new punto();
25     llenar(a,llenar_nombre[0],llenar_x[0],llenar_y[0]);
26     arbol_m.vecinos_mas_cercanos(a,37000);
27     cout<<"
28     llenar(a,llenar_nombre[5],llenar_x[5],llenar_y[5]);
29     arbol_m.vecinos_mas_cercanos(a);
30     /*
31     cout<<"
32     m_tree arbol_n;
33     string llenar_nombre2[20] = {"USA","Slovenia","Slovakia","San_Marino","Romania","Peru", "North_Macedonia","Montene
34     ,"Lithuania","Hungary","Gibraltar","Georgia","Czechia","Croatia","Colombia","Bulgaria","Brazil","Bosnia_and_Herzeg
35     ,"Armenia","Argentina"};
36     float llenar_x2[20] = {148104,202419,124480,175688,93390,66477,103485,250528,176235,114600,215221
37     ,212561,200245,149455,98156,101106,102912,84584,113938,116439};
38     float llenar_y2[20] = {2406,2512,2639,2733,2966,5983,3639,3673,2525,3586,2910,3030,3080,2678,2489
39     ,4139,2863,3870,2547,2547};
40     for (int i = 0; i < 20; i++){
41         punto *a = new punto();
42         llenar(a,llenar_nombre2[i],llenar_x2[i],llenar_y2[i]);
43         arbol_n.anadir(a);
44         delete a;
45         if(i == 4 || i == 9){
46             cout<<arbol_n;
47             cout<<"
```

Lín. 29, Col. 5 (32 seleccionada) Espacios: 4 UTF-8 CRLF C++ Win32

```

25     llenar(a, llenar_nombre[0], llenar_x[0], llenar_y[0]);
26     arbol_m.vecinos_mas_cercanos(a, 37000);
27     cout<<"_____ "<<endl;
28     llenar(a, llenar_nombre[5], llenar_x[5], llenar_y[5]);
29     arbol_m.vecinos_mas_cercanos(a);
30     /*-----*/
31     cout<<"_____ "<<endl;

```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN **TERMINAL** JUPYTER

```

G -> Romania;
H -> Gibraltar;
I -> Peru;
I -> Bulgaria;
I -> Bosnia_and_Herzegovina;
I -> Montenegro;
N -> Hungary;
N -> San_Marino;
N -> Croatia;
N -> Slovakia;
Q -> O;
O -> Argentina;
O -> Armenia;

```

No hay vecino cercano en dicho rango

Armenia

Lín. 29, col. 37 Espacios: 4 UTF-8 CRLF C++ Win32 23:00 26/11/2022

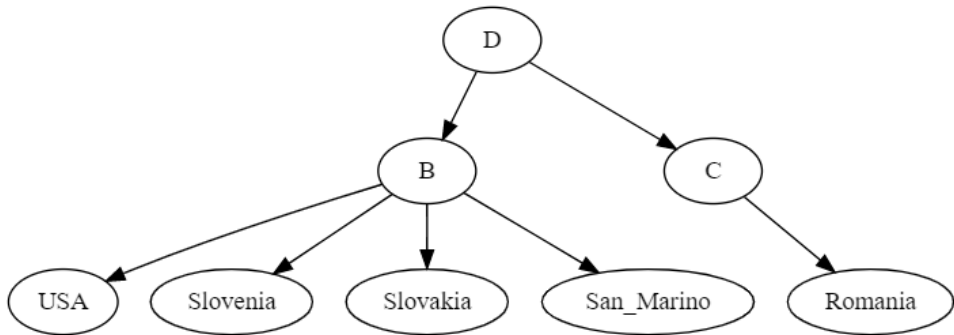
5. Cuál de las coordenadas (x o y) es la más determinante en el cálculo de las distancias?

Es aquella que tenga mayor variación, en este caso es “x”.

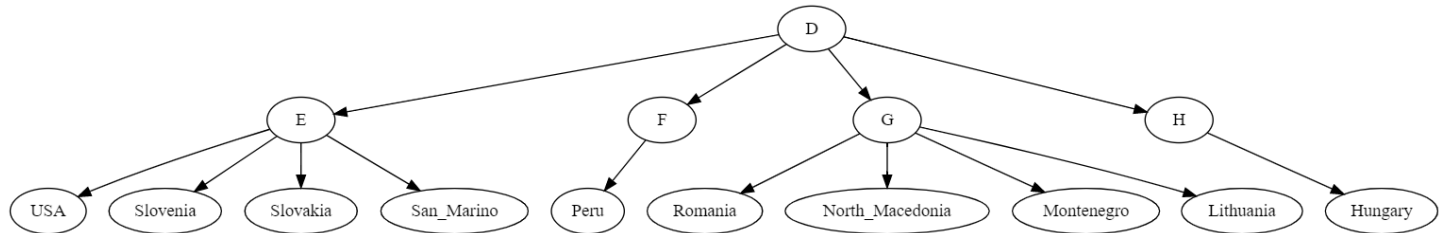
6. Utilice el conjunto de datos proporcionado en la tabla 2 para determinar si existe influencia del orden de inserción de los datos en el resultado de las búsquedas en términos de los nodos recorridos para generar la respuesta.

Ejercicio 2:

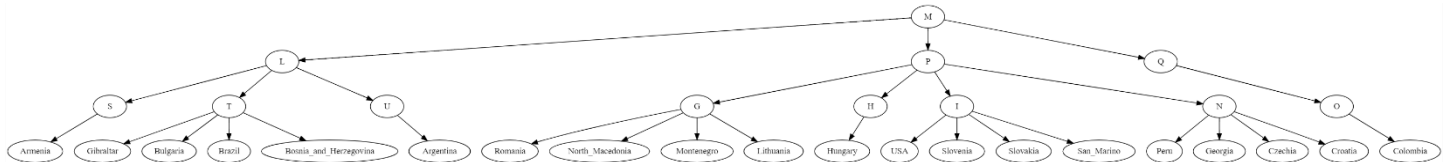
a) El quinto elemento de la tabla (North Macedonia)



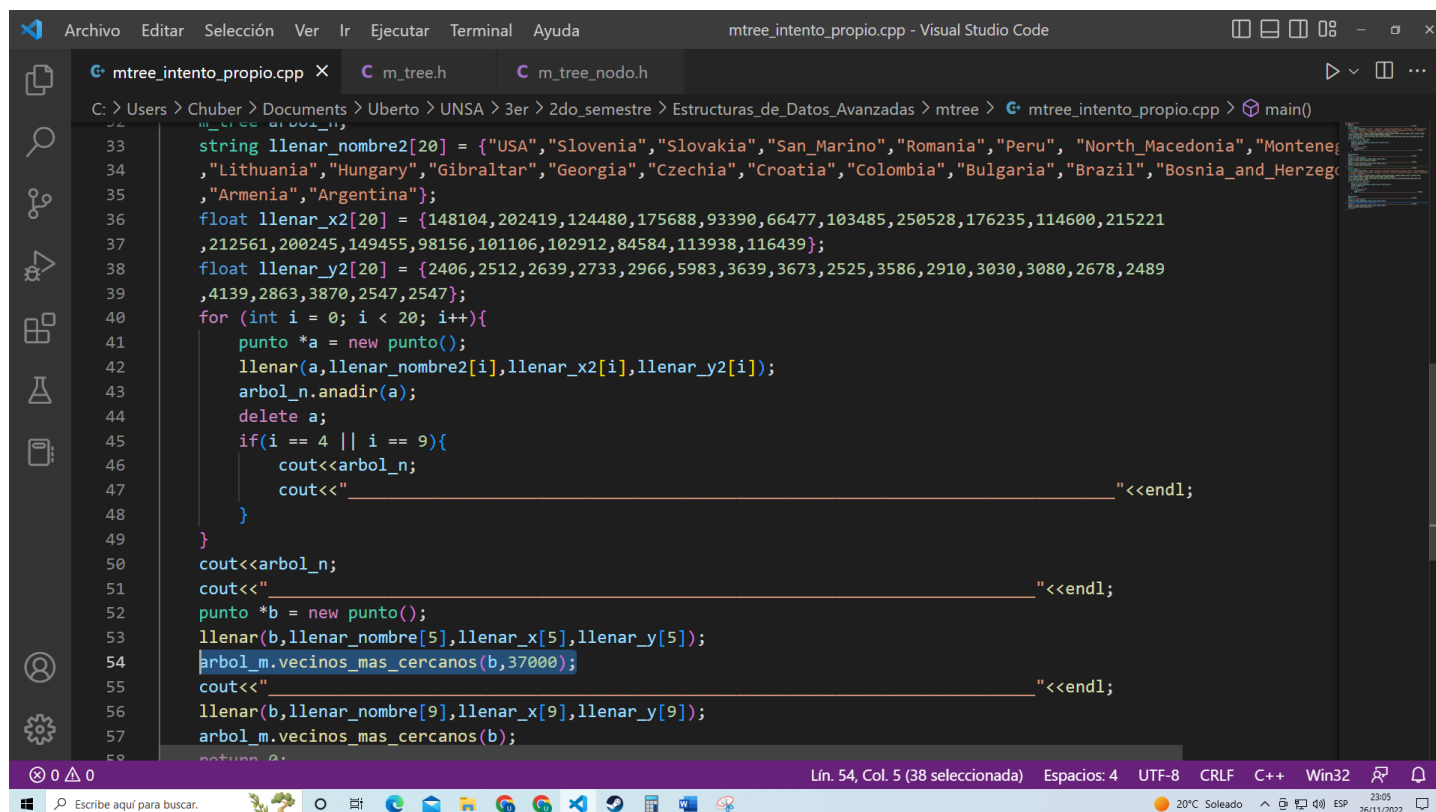
b) El décimo elemento de la tabla (Gibraltar)



c) El último elemento de la tabla (USA)

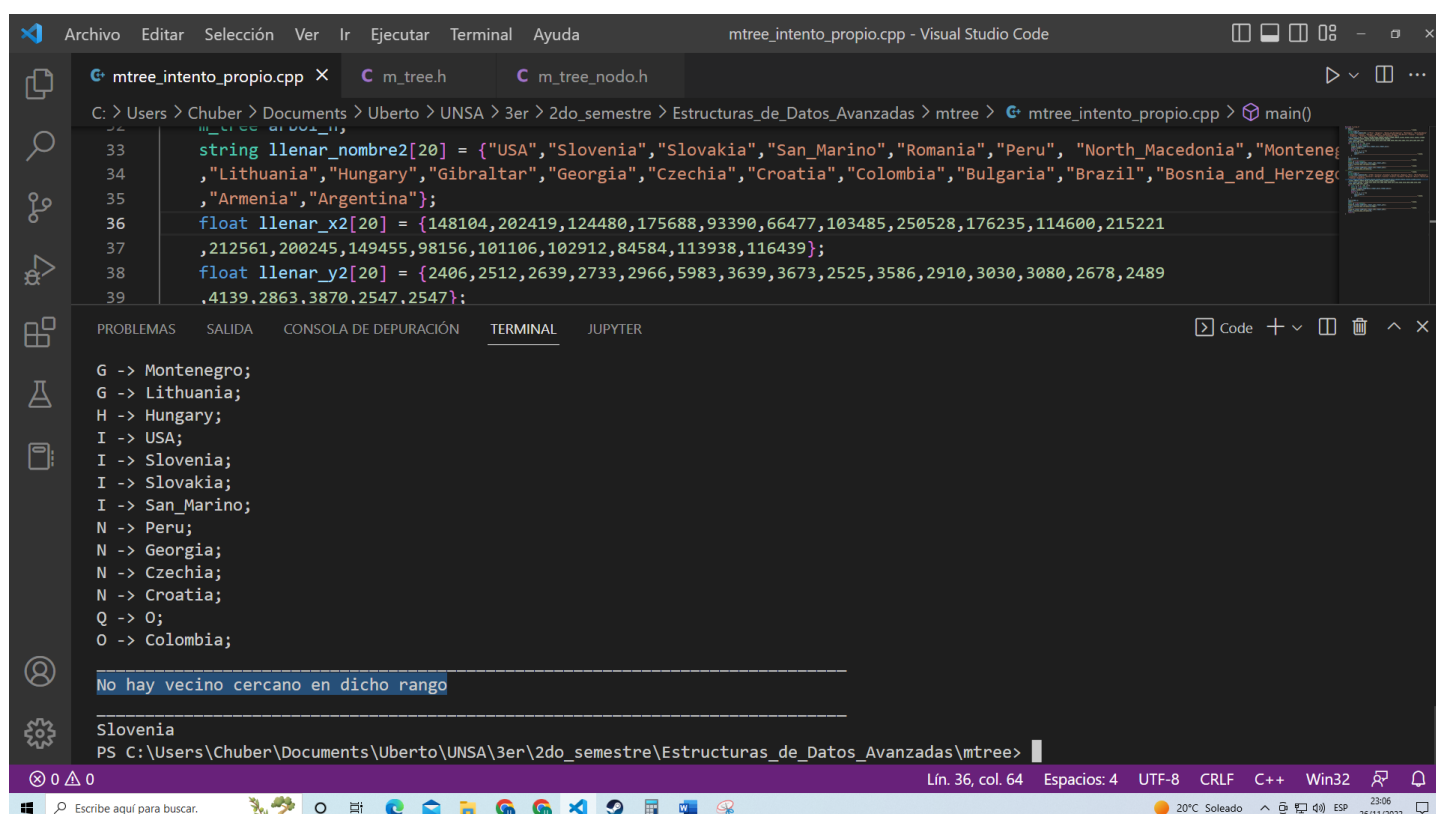


Ejercicio 3:



The screenshot shows the Visual Studio Code editor with the file `mtree_intento_propio.cpp` open. The code defines a binary tree structure with a `punto` struct and a `arbol` struct. The `main` function populates the tree with 20 nodes, each containing a country name and coordinates. The tree is then traversed to find the closest neighbor to a specific point.

```
33 string llenar_nombre2[20] = {"USA", "Slovenia", "Slovakia", "San_Marino", "Romania", "Peru", "North_Macedonia", "Montenegro",
34 , "Lithuania", "Hungary", "Gibraltar", "Georgia", "Czechia", "Croatia", "Colombia", "Bulgaria", "Brazil", "Bosnia_and_Herzegovina",
35 , "Armenia", "Argentina"};
36 float llenar_x2[20] = {148104, 202419, 124480, 175688, 93390, 66477, 103485, 250528, 176235, 114600, 215221,
37 , 212561, 200245, 149455, 98156, 101106, 102912, 84584, 113938, 116439};
38 float llenar_y2[20] = {2406, 2512, 2639, 2733, 2966, 5983, 3639, 3673, 2525, 3586, 2910, 3030, 3080, 2678, 2489,
39 , 4139, 2863, 3870, 2547, 2547};
40 for (int i = 0; i < 20; i++){
41     punto *a = new punto();
42     llenar(a, llenar_nombre2[i], llenar_x2[i], llenar_y2[i]);
43     arbol_n.anadir(a);
44     delete a;
45     if(i == 4 || i == 9){
46         cout<<arbol_n;
47         cout<<"\n";
48     }
49 }
50 cout<<arbol_n;
51 cout<<"\n";
52 punto *b = new punto();
53 llenar(b, llenar_nombre2[5], llenar_x2[5], llenar_y2[5]);
54 arbol_m.vecinos_mas_cercanos(b, 37000);
55 cout<<"\n";
56 llenar(b, llenar_nombre2[9], llenar_x2[9], llenar_y2[9]);
57 arbol_m.vecinos_mas_cercanos(b);
58 return 0;
```



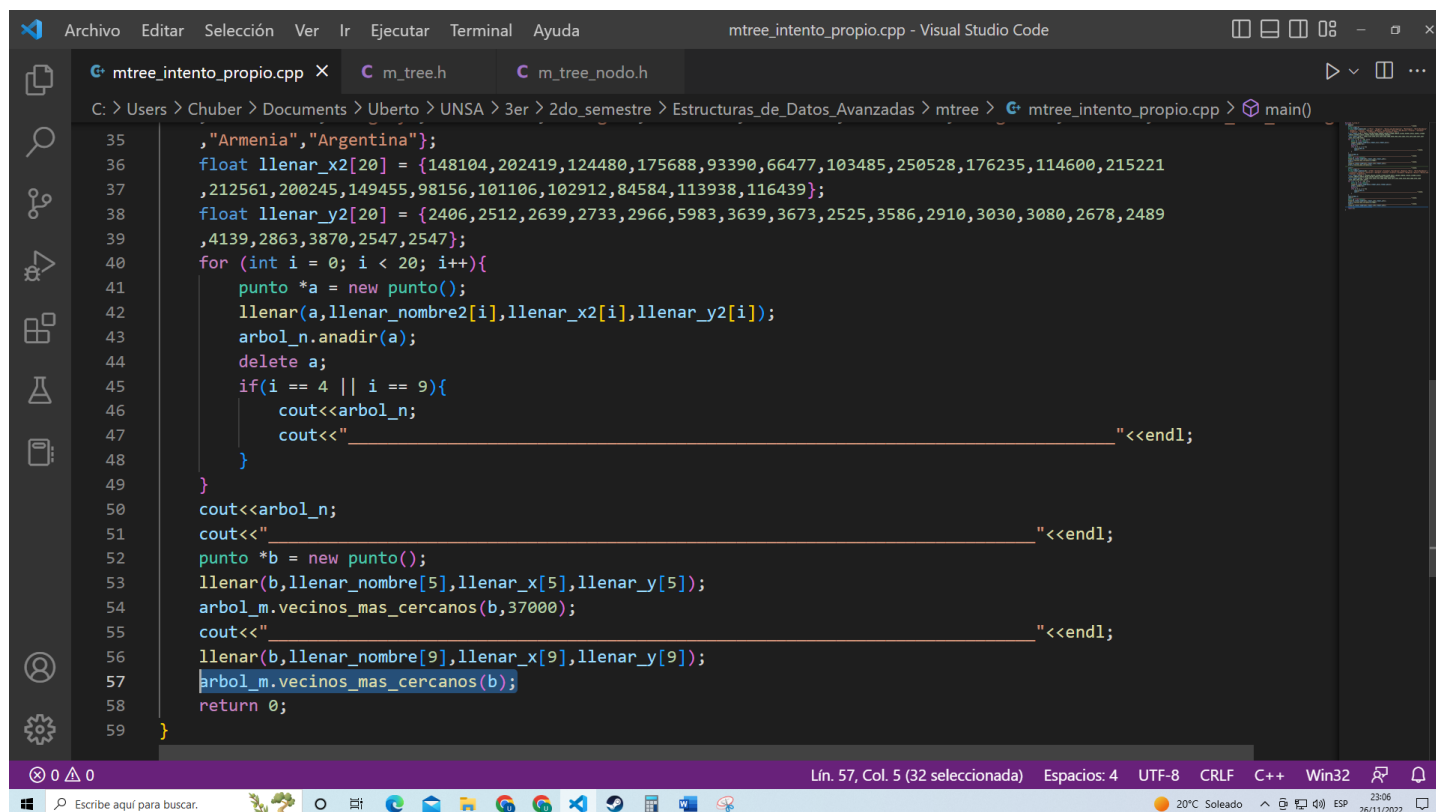
The screenshot shows the Visual Studio Code editor with the file `mtree_intento_propio.cpp` open. The terminal window is active, displaying the output of the program. The output shows the tree structure for the first 10 nodes, followed by a message indicating that no neighbor was found within the specified range for the point at index 5.

```
G -> Montenegro;
G -> Lithuania;
H -> Hungary;
I -> USA;
I -> Slovenia;
I -> Slovakia;
I -> San_Marino;
N -> Peru;
N -> Georgia;
N -> Czechia;
N -> Croatia;
Q -> 0;
O -> Colombia;

No hay vecino cercano en dicho rango

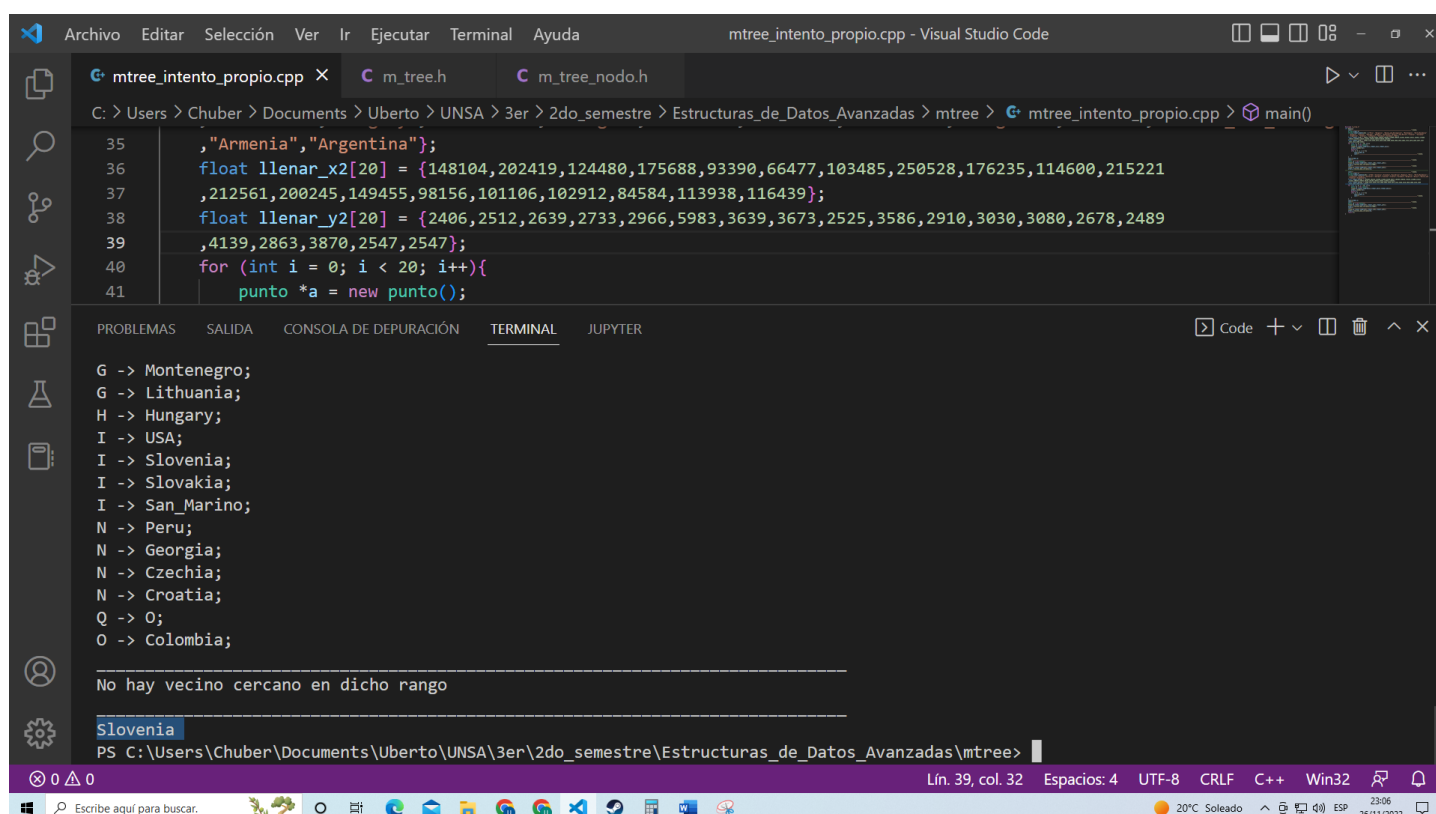
Slovenia
PS C:\Users\Chuber\Documents\Uberto\UNSA\3er\2do_semestre\Estructuras_de_Datos_Avanzadas\mtree>
```

Ejercicio 4:



The screenshot shows the Visual Studio Code editor with the file `mtree_intento_propio.cpp` open. The code is in C++ and implements a binary tree structure. The tree is populated with data for various countries, including Armenia, Argentina, Montenegro, Lithuania, Hungary, USA, Slovenia, Slovakia, San Marino, Peru, Georgia, Czechia, Croatia, and Colombia. The code uses arrays to store the data and a recursive function to traverse the tree. The output of the program is shown in the terminal window at the bottom.

```
35  , "Armenia", "Argentina");
36  float llenar_x2[20] = {148104, 202419, 124480, 175688, 93390, 66477, 103485, 250528, 176235, 114600, 215221
37  , 212561, 200245, 149455, 98156, 101106, 102912, 84584, 113938, 116439};
38  float llenar_y2[20] = {2406, 2512, 2639, 2733, 2966, 5983, 3639, 3673, 2525, 3586, 2910, 3030, 3080, 2678, 2489
39  , 4139, 2863, 3870, 2547, 2547};
40  for (int i = 0; i < 20; i++){
41      punto *a = new punto();
42      llenar(a, llenar_nombre2[i], llenar_x2[i], llenar_y2[i]);
43      arbol_n.anadir(a);
44      delete a;
45      if(i == 4 || i == 9){
46          cout<<arbol_n;
47          cout<<"_____<<endl;
48      }
49  }
50  cout<<arbol_n;
51  cout<<"_____<<endl;
52  punto *b = new punto();
53  llenar(b, llenar_nombre[5], llenar_x[5], llenar_y[5]);
54  arbol_m.vecinos_mas_cercanos(b, 37000);
55  cout<<"_____<<endl;
56  llenar(b, llenar_nombre[9], llenar_x[9], llenar_y[9]);
57  arbol_m.vecinos_mas_cercanos(b);
58  return 0;
59 }
```



The screenshot shows the Visual Studio Code editor with the file `mtree_intento_propio.cpp` open. The code is in C++ and implements a binary tree structure. The tree is populated with data for various countries, including Armenia, Argentina, Montenegro, Lithuania, Hungary, USA, Slovenia, Slovakia, San Marino, Peru, Georgia, Czechia, Croatia, and Colombia. The code uses arrays to store the data and a recursive function to traverse the tree. The output of the program is shown in the terminal window at the bottom.

```
35  , "Armenia", "Argentina");
36  float llenar_x2[20] = {148104, 202419, 124480, 175688, 93390, 66477, 103485, 250528, 176235, 114600, 215221
37  , 212561, 200245, 149455, 98156, 101106, 102912, 84584, 113938, 116439};
38  float llenar_y2[20] = {2406, 2512, 2639, 2733, 2966, 5983, 3639, 3673, 2525, 3586, 2910, 3030, 3080, 2678, 2489
39  , 4139, 2863, 3870, 2547, 2547};
40  for (int i = 0; i < 20; i++){
41      punto *a = new punto();
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL JUPYTER

```
G -> Montenegro;
G -> Lithuania;
H -> Hungary;
I -> USA;
I -> Slovenia;
I -> Slovakia;
I -> San_Marino;
N -> Peru;
N -> Georgia;
N -> Czechia;
N -> Croatia;
Q -> 0;
O -> Colombia;

No hay vecino cercano en dicho rango

Slovenia
PS C:\Users\Chuber\Documents\Uberto\UNSA\3er\2do_semestre\Estructuras_de_Datos_Avanzadas\mtree>
```