Lab 9. For each. Structured bindings.

Write a C++ template **Map** that will make the following code :

```
int main()
{
    Map<int, const char *> m;
    m[10] = "C++";
    m[20] = "test";
    m[30] = "Poo";
    for (auto[key, value, index] : m)
    {
        printf("Index:%d, Key=%d, Value=%s\n", index, key, value);
    }
    m[20] = "result";
    for (auto[key, value, index] : m)
    {
        printf("Index:%d, Key=%d, Value=%s\n", index, key, value);
    }
    return 0;
}
```

print the following on the screen:

```
Index:0, Key=10, Value=C++
Index:1, Key=20, Value=test
Index:2, Key=30, Value=Poo
Index:0, Key=10, Value=C++
Index:1, Key=20, Value=result
Index:2, Key=30, Value=Poo
```

For this task, use the following:

- structured binding
- auto keyword
- for-each

Besides this - add the following methods:

- a method **Set** that can be used to associate a value to a key
- a method **Get** using the following syntax ***bool Get(const K& key, V& value)*** that will copy the value associated to parameter **key** into the parameter **value** and returns **true**. If the parameter **key** does not exists in the map, **Get** method will return **false**
- a method **Count** that returns the amount of elements in the map
- a method **Clear** that clears the entire map
- a method **Delete** that deletes a specific key (if exists) --> use the following syntax: *bool Delete(const K& key)*
- a method **Includes** with the following syntax: ***bool Includes(const Map<K,V>& map)*** that checks if a map is included in another. A map A is included in another map B,

if all keys from map A exists in map B

YOU ARE NOT ALLOWED TO USE STL TEMPLATES (Vector, Map, List, etc) for this problem.