Tutorial 73 - Map in C++ STL

Introduction to Maps in C++ STL

- A **Map** is an **associative container** in C++ STL that stores elements in **key-value pairs**.
- It automatically **sorts the elements based on the keys**.
- **Example Use Case**: Storing students' names as **keys** and their marks as **values**.

Declaring and Using Maps

Syntax for Declaring a Map

```
map<data_type_of_key, data_type_of_value> variable_name;

```

Example:

```
map<string, int> marksMap;

```

Basic Operations on Maps

1. Storing Key-Value Pairs

- **Using Indexing Method**:

```
marksMap["Atul"] = 58;
marksMap["Rohit"] = 57;
marksMap["Kishlay"] = 78;
marksMap["Aditya"] = 65;
marksMap["Sachin"] = 53;

```

2. Iterating Through a Map

- **Using Iterator & Looping**

```
#include<iostream>
#include<map>
#include<string>

using namespace std;

int main() {
    map<string, int> marksMap;
    marksMap["Atul"] = 58;
    marksMap["Rohit"] = 57;
    marksMap["Kishlay"] = 78;
    marksMap["Aditya"] = 65;
    marksMap["Sachin"] = 53;

    map<string, int>::iterator iter;
    for (iter = marksMap.begin(); iter != marksMap.end(); iter++) {
        cout << (*iter).first << " " << (*iter).second << "\n";
```

```
    }

    return 0;
}

```

Output:

```
Aditya 65
Atul 58
Kishlay 78
Rohit 57
Sachin 53

```

📌 **Note:** The map automatically sorts elements by **keys**.

---

3. Inserting Elements Using `insert()`

Syntax

```
marksMap.insert({ {"Rohan", 89}, {"Akshat", 46} });

```

Program Using `insert()`

```
marksMap.insert({ {"Rohan", 89}, {"Akshat", 46} });

```

Output:

```
Aditya 65
Akshat 46
Atul 58
Kishlay 78
Rohan 89
Rohit 57
Sachin 53

```

Additional Map Methods

Method	Description
`size()`	Returns the number of key-value pairs in the map.
`empty()`	Checks if the map is empty (returns `true` or `false`).
`erase(key)`	Removes a key-value pair using the key.
`clear()`	Removes all elements from the map.

Short Notes on Maps in C++ STL

- **Map is an associative container** that stores key-value pairs.
- **Keys are unique**, and elements are **sorted in ascending order** of keys.
- **Ways to insert elements:**
  - **Indexing method** (`marksMap["key"] = value`)
  - `insert()` **method** (`marksMap.insert({{"key", value}})`)
- **Accessing elements:**
  - **Using iterators** with `.begin()` and `.end()`.
  - **Using** `.first` **and** `.second` to access key and value.
- **Useful functions:**
  - `size()`, `empty()`, `erase()`, `clear()`.

Conclusion

- **Maps are useful** for storing key-value pairs and allow **fast retrieval** of values based on keys.
- **Practice more by exploring STL documentation**: std::map - C++ Reference
- Next, we will learn about **unordered maps in C++ STL**. 🚀