* C:\Coading\Supreme3.0\04\_Week\04\_C++\_STL\*pdf
* C:\Coading\Supreme3.0\04\_Week\04\_C++\_STL\*
* C:\Coading\Supreme3.0\04\_Week\04\_Mega\_S&S\*
* C:\Coading\Supreme3.0\04\_Week\04\_Searching\_and\_Sorting\*
* Assignment 1. [**532. K-diff Pairs in an Array**](https://leetcode.com/problems/k-diff-pairs-in-an-array/)

2. [**658. Find K Closest Elements**](https://leetcode.com/problems/find-k-closest-elements/)

3.exponontail search

4.allocate minimum number of pages (gfg)

5. **The Painter's Partition Problem-II (gfg)**

6. Aggressive Cows(gfg)

7. <https://www.spoj.com/problems/EKO/>

8. <https://www.spoj.com/problems/PRATA/> ( not understood )

* selection,insertion,bubbleSort by love babbar\*
* Sorting algorithm pdf(dipankar)\*

| **Container** | **Supports Iterators? (begin(), end())** | **Alternative Way** |
| --- | --- | --- |
| vector | ✅ Yes | Iterators work |
| deque | ✅ Yes | Iterators work |
| list | ✅ Yes | Iterators work |
| set / map | ✅ Yes | Iterators work |
| queue | ❌ No | Use front() and pop() in a loop |
| stack | ❌ No | Use top() and pop() in a loop |
| priority\_queue | ❌ No | Use top() and pop() in a loop |

**Does a[i].swap(a[j]) Work in queue, stack, and priority\_queue?**

No, **a[i].swap(a[j]) will NOT work** for queue, stack, or priority\_queue.

**🔴 Why Doesn't a[i].swap(a[j]) Work?**

* **Queue, Stack, and Priority Queue** do **NOT provide direct access** to elements using indexing (a[i]).
* They are **sequential containers** that only allow **push/pop** operations.
* The swap() function works **only for STL containers like vector, list, or deque**, where you have

**Accessing Keys in std::map in C++**

There are **multiple ways** to access the **keys** of a map in C++:

**1️⃣ Using a for loop with .first**

Every element in a map is a pair<Key, Value>, so you can access the **key** using .first.

cpp

CopyEdit

#include <iostream>

#include <map>

using namespace std;

int main() {

map<int, string> students = {{101, "Alice"}, {102, "Bob"}, {103, "Charlie"}};

cout << "Keys in the map: ";

for (auto it : students) {

cout << it.first << " "; // Accessing the key

}

return 0;

}

🔹 **Output:**

arduino

CopyEdit

Keys in the map: 101 102 103

**2️⃣ Using an Iterator (it->first)**

If you want to use an **iterator**, access the key using it->first instead of it.first.

cpp

CopyEdit

#include <iostream>

#include <map>

using namespace std;

int main() {

map<int, string> students = {{101, "Alice"}, {102, "Bob"}, {103, "Charlie"}};

cout << "Keys using iterator: ";

for (map<int, string>::iterator it = students.begin(); it != students.end(); ++it) {

cout << it->first << " "; // Accessing the key

}

return 0;

}

🔹 **Output:**

vbnet

CopyEdit

Keys using iterator: 101 102 103

**3️⃣ Using for\_each() with Lambda Function (C++11 and above)**

If you prefer **modern C++**, you can use for\_each() with a lambda function.

cpp

CopyEdit

#include <iostream>

#include <map>

#include <algorithm>

using namespace std;

int main() {

map<int, string> students = {{101, "Alice"}, {102, "Bob"}, {103, "Charlie"}};

cout << "Keys using for\_each: ";

for\_each(students.begin(), students.end(), [](const pair<int, string> &p) {

cout << p.first << " ";

});

return 0;

}

🔹 **Output:**

cpp

CopyEdit

Keys using for\_each: 101 102 103

**4️⃣ Extracting Keys into a vector**

If you need to **store all keys separately** in a vector, you can do:

cpp

CopyEdit

#include <iostream>

#include <map>

#include <vector>

using namespace std;

int main() {

map<int, string> students = {{101, "Alice"}, {102, "Bob"}, {103, "Charlie"}};

vector<int> keys;

for (auto it : students) {

keys.push\_back(it.first);

}

cout << "Keys in vector: ";

for (int key : keys) {

cout << key << " ";

}

return 0;

}

🔹 **Output:**

cpp

CopyEdit

Keys in vector: 101 102 103

**🔴 Summary**

| **Method** | **How to Access the Key** |
| --- | --- |
| Range-based for loop | it.first |
| Iterator (map<int, string>::iterator) | it->first |
| for\_each() with lambda | p.first inside lambda |
| Store keys in a vector | Use push\_back(it.first) |

✅ **Use .first for direct access, it->first for iterators, and lambdas for modern C++.** 🚀