



Lecture 26

Part 2: Unordered Maps & Multimaps

- **unordered_map complexity** : in any case either its find , put , get ,anything is $O(1)$;

thats way all time we use **unordered_map**

- Unordered_map as key not allowed container's like `map<<vector> , int>` is not allowed .
- **multimap** : it is used only that we store the duplicate key's in the map .

```

9
10 int main(){
11     // 1. inbuilt implementation
12     // 2. Time complexity
13     // 3 . valid keys datatype
14     unordered_map<int, string > m;
15     m[1] = "abc"; // O(1)
16     m[5] = "cdc";
17     m[3] = "acd";
18     m[6] = "a";
19     m[5] = "cde";
20     auto it = m.find(7); // O(1)
21     if(it != m.end())
22         m.erase(it); // log(1)
23     // m.clear();
24     // if(it == m.end()){
25     //     cout << "NO value";
26     // }else{
27     //     cout << (*it).first << " " << (*it).second
28     // }

```

[Finished in 2.4s]

input.in

```

1 8
2 abc
3 def
4 abc
5 ghj
6 jkl
7 ghj
8 ghj
9 abc

```

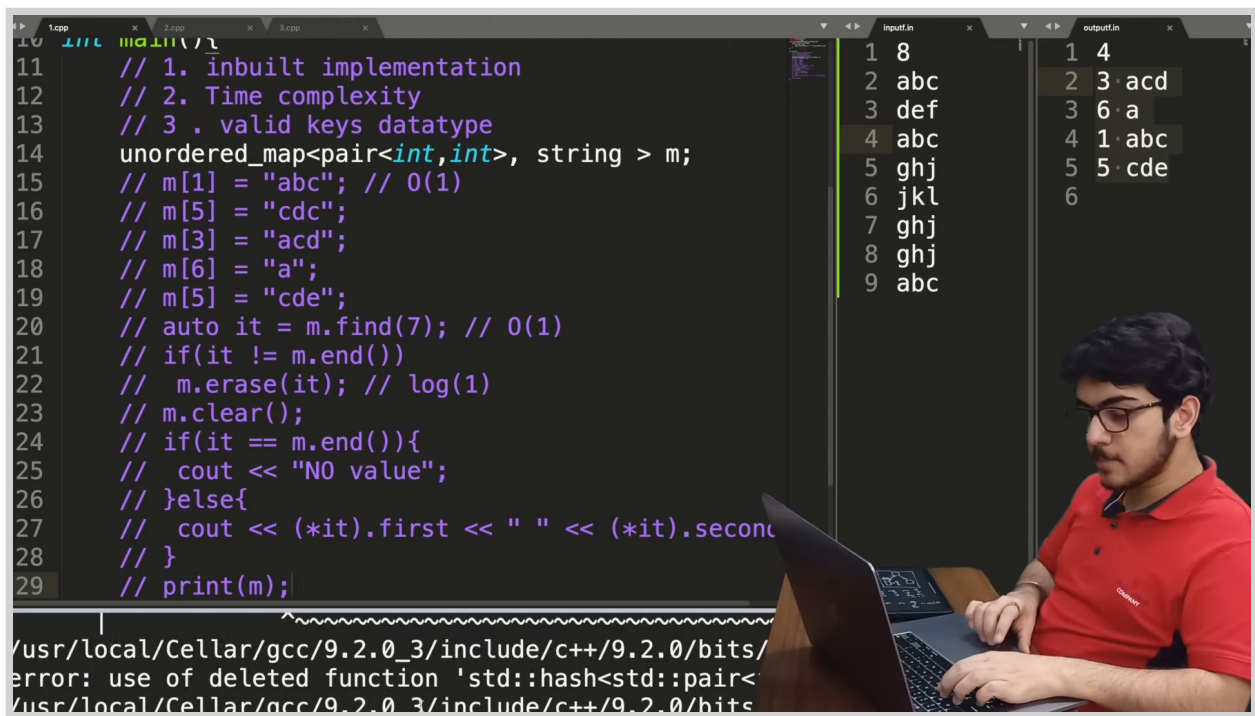
output.in

```

1 4
2 3 acd
3 6 a
4 1 abc
5 5 cde
6

```

container's not allowed as a key.



```

10 // 1. inbuilt implementation
11 // 2. Time complexity
12 // 3 . valid keys datatype
13 unordered_map<pair<int,int>, string > m;
14 // m[1] = "abc"; // 0(1)
15 // m[5] = "cdc";
16 // m[3] = "acd";
17 // m[6] = "a";
18 // m[5] = "cde";
19 // auto it = m.find(7); // 0(1)
20 // if(it != m.end())
21 // m.erase(it); // log(1)
22 // m.clear();
23 // if(it == m.end()){
24 // cout << "NO value";
25 // }else{
26 // cout << (*it).first << " " << (*it).second;
27 // }
28 // print(m);
29
/usr/local/Cellar/gcc/9.2.0_3/include/c++/9.2.0/bits/
error: use of deleted function 'std::hash<std::pair<
/usr/local/Cellar/gcc/9.2.0_3/include/c++/9.2.0/bits

```

Input.in

```

1 8
2 abc
3 def
4 abc
5 ghj
6 jkl
7 ghj
8 ghj
9 abc

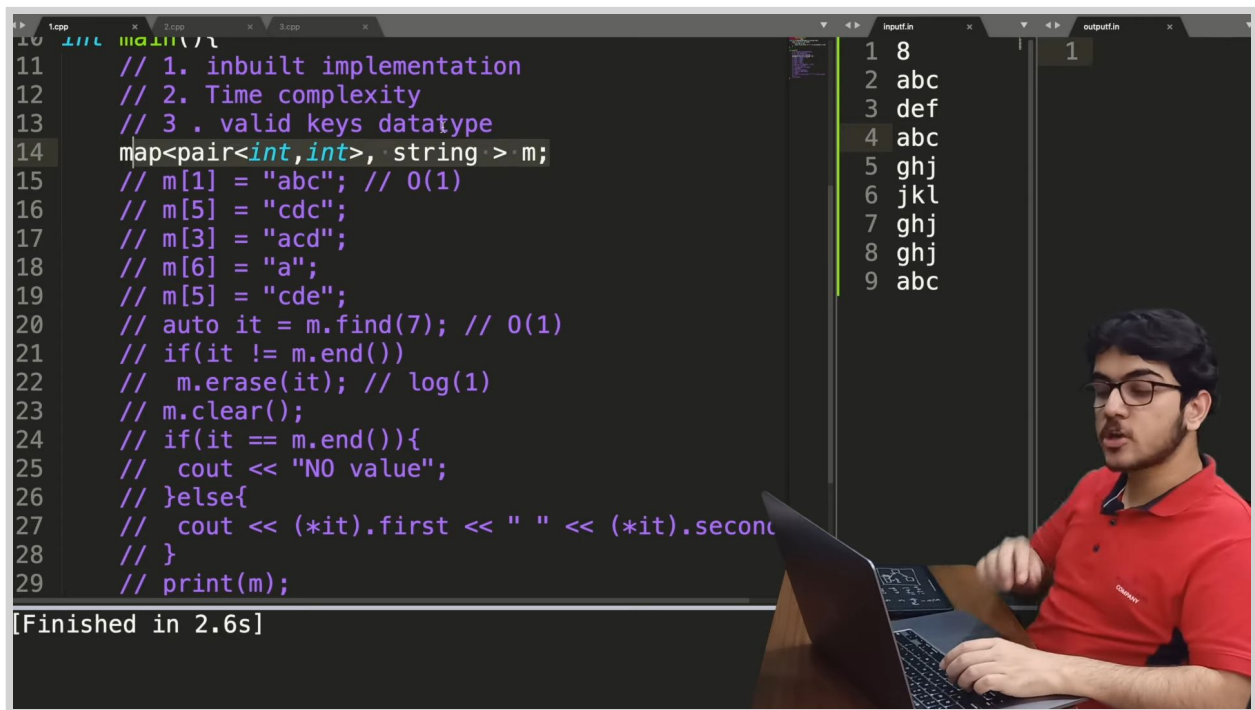
```

output.in

```

1 4
2 3 acd
3 6 a
4 1 abc
5 5 cde
6

```



```

10 // 1. inbuilt implementation
11 // 2. Time complexity
12 // 3 . valid keys datatype
13 map<pair<int,int>, string > m;
14 // m[1] = "abc"; // 0(1)
15 // m[5] = "cdc";
16 // m[3] = "acd";
17 // m[6] = "a";
18 // m[5] = "cde";
19 // auto it = m.find(7); // 0(1)
20 // if(it != m.end())
21 // m.erase(it); // log(1)
22 // m.clear();
23 // if(it == m.end()){
24 // cout << "NO value";
25 // }else{
26 // cout << (*it).first << " " << (*it).second;
27 // }
28 // print(m);
29
[Finished in 2.6s]

```

Input.in

```

1 8
2 abc
3 def
4 abc
5 ghj
6 jkl
7 ghj
8 ghj
9 abc


```

output.in

```

1

```



```

8 }
9
10 int main(){
11     // 1. inbuilt implementation
12     // 2. Time complexity
13     // 3 . valid keys datatype
14     multimap<pair<int,int>, string > m;
15
16     // m[1] = "abc"; // 0(1)
17     // m[5] = "cdc";
18     // m[3] = "acd";
19     // m[6] = "a";
20     // m[5] = "cde";
21     // auto it = m.find(7); // 0(1)
22     // if(it != m.end())
23     //     m.erase(it); // log(1)
24     // m.clear();
25     // if(it == m.end()){
26     //     cout << "NO value";
27     // }else{

```

[Finished in 3.1s]

Input.in

```

1 8
2 abc
3 def
4 abc
5 ghj
6 jkl
7 ghj
8 ghj
9 abc
10 2
11 abc
12 ghj

```

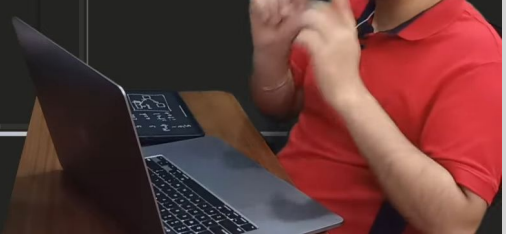
output.in

```

1 3
2 3
3

```

that's way to replace the multimap



```

8 }
9
10 int main(){
11     // 1. inbuilt implementation
12     // 2. Time complexity
13     // 3 . valid keys datatype
14     map<int, vector<string> > m;
15
16     // m[1] = "abc"; // 0(1)
17     // m[5] = "cdc";
18     // m[3] = "acd";
19     // m[6] = "a";
20     // m[5] = "cde";
21     // auto it = m.find(7); // 0(1)
22     // if(it != m.end())
23     //     m.erase(it); // log(1)
24     // m.clear();
25     // if(it == m.end()){
26     //     cout << "NO value";
27     // }else{

```

[Finished in 3.1s]

Input.in

```

1 8
2 abc
3 def
4 abc
5 ghj
6 jkl
7 ghj
8 ghj
9 abc
10 2
11 abc
12 ghj

```

output.in

```

1 3
2 3
3

```

