C, C++, DSA in depth

map

Saurabh Shukla (MySirG)

Agenda

1. map
2. Creating map object
3. Accessing map elements
4. Implicit and explicit iterators
5. map methods

map

It is used for the storage and retrieval of data from a collection in which each element is a pair that has both a data value and a sort key.

The value of the key is unique and is used to automatically sort the data.

The value in a map can be changed directly.

The key can't be changed, instead key associated with value must be deleted and new key-value must be inserted.

It provides a bidirectional Iterator

It is based on RBT

header <map> has to be included

How to create a map object?

```
map <int, string> m1 = { {1, "Bhopal"}, {2, "Indore"},
                         {3, "Pune"} };

map <int, string> m2;
```

Accessing map elements

You can access map in the variety of ways.

1. implicit iterator
2. explicit iterator

Implicit Iterator

```
map <int, string> m1 = { {1, "Bhopal"}, {2, "Indore"},
                         {3, "Pune"} };

for (auto p:m1)
{
    cout<<p.first<<"  "<<p.second<<endl;

}
```

Explicit iterator

You can get an iterator object from the following members

①	begin()	end()	iterator
②	cbegin()	cend()	const_iterator
③	rbegin()	rend()	reverse_iterator
④	crbegin()	crend()	const_reverse_iterator

Explicit Iterator

```
map <int, string> m1 = { {1, "Bhopal"}, {2, "Indore"},
                         {3, "Pune"} };

map <int, string> :: iterator it;

for ( it = m1.begin(); it != m1.end(); it++)
    cout << it -> first << "  " << it -> second;
```

Methods of map

at()
count()
empty()

emplace()
insert()

clear()
erase()

swap()
size()