Assignment No 12

Name :→ Sameer Manoj Brambecha
Roll No :→ 21115
Batch :→ E-1
Date of performance :→ 7/12/2021
Date of Submission :→ 10/12/2021

Title :→ Map Associative Container.

Problem Statement :→
Write a program in C++ to use map associative container. The keys will be the names of states and the values will be the population of the states. When the program runs, the user is prompted to type the name of a state. The program then looks in the map, using the state name as an index and returns the population of the state.

Learning Objectives :→
To learn the concept of map associative container.

Learning outcomes:
After implementation of this assignments, students will be able to ~~learn the use of~~ implement map associative container and functions related to it.

S/W and H/W requirements:
1> 64 bit Windows 10 O.S
2.) Open source C++ programming tool like G++/Gcc.

Theory :→

Map :→ The map is an associative container in which the elements are stored in the form of key value and mapped value. For example - {(1, a), (2, b), (3, c)}. In a map, the key values are generally used to sort and uniquely identify the elements, while the mapped values store the content associated to this key. The types of key and mapped value may differ. The duplicate values are not allowed in map. Hence it possess one to one relationship. The header file <map> is included in the program for creating and using the map object.

The syntax for creating map is

map<key, value> map_name;

Various operations that can be performed on map are :→

	Function Name	Purpose
1.)	insert (pair)	The element is inserted into the map.
2.)	erase (iterator it)	The element pointed by the iterator is deleted from the map.
3.)	void swap (map)	Swaps the contents of the map.
4.)	void clear ()	Clears the content.
5.)	size_type size()	Returns the size of container.

Iterators:→

The iterators are basically objects but sometimes they can be pointers and hence iterators specify the positions in containers. The iterators are used to traverse the contents of container.

The operators used for iterating through the container are enlisted below:—

	Operator	Purpose
1	++	Make the iterator step forward to the next element.
2	== and !=	Return whether two iterators represent the same position or not.
3	=	Assigns an iterator.

→ The most commonly used functions for iterating through container are:

1> begin() :→ Returns an iterator representing the beginning of the elements in the container.

2> end() :→ Returns an iterator representing the element just past the end of the elements.

→ Iterator can be specified in two ways:→

1> container::iterator provides a read/write iterator.

2> container::const_iterator provides a read only iterator.

Algorithm:→

I) Algorithm for class Map.

1) Start.
2) Include header files such as 'iostream', 'map', 'string', and 'utility'.
3.) Define class Map
4.) Declare variables like n (integer), name (string) and p (long).
5.) In public part, initialize map m1 and type of key, value pair as (string, long).
6.) Declare functions insert(), display() and search().
7.) Stop.

II.) Algorithm for insert(-) :→

1.) Start.
2.) Read no. of ~~steps~~ states from the user 'n'
3.) Read the name of state and population of state one by one.
4.) Using m1.insert(pair<string, long>(name, p));
insert the states and their population in map.
5.) Repeat steps 3 and 4 'n' no. of times.
6.) Stop.

III.) Algorithm for display() :→

1.) Start.
2.→ ~~Init~~ Declare iterator it.
3.> ~~Write~~ Display name of state and its population using iterator.

4.) Repeat ~~iterator~~ step 3 till the iterator reaches end of container.
5.) Stop.

IX> Algorithm for search() :→

1> Start.
2> Declare iterator it.
3> Declare string 's'.
4.) Read name of state whose population is to be displayed.
5.) Search 's' using m.find(s)
6.) ~~If~~ Search until iterator reaches end of container.
7.) If found then display the population.
8.) Stop.

X) Algorithm for main function:—

1.) Start
2.) Create object of class Map i.e 'm'.
3.) Display Menu with 4 options 1) Insert; 2) Display, 3) Search 4) Exit.
4.) Read User's choice.
5.) According to user's choice, call respective function.
6) Repeat step 5 until user enters choice as '4'.
7) Stop.

Test Cases :→

Test Case No	Test Case Description	Input	Expected Output	Actual Output	Result
1.	Insert & Display	name = Goa p = 10,00,000 name = Maharashtra p = 100000000	Goa: 10,00,000 Maharashtra: 100000000	Goa: 10,00,000 Maharashtra: 100000000.	Pass.
2.	Search.	s = Goa.	Goa: 1000000	Goa: 10,00,000	Pass.

Conclusion :→

Hence we have successfully studies the concept of map associative container.