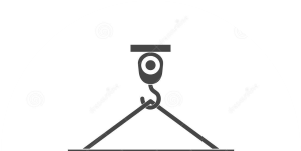
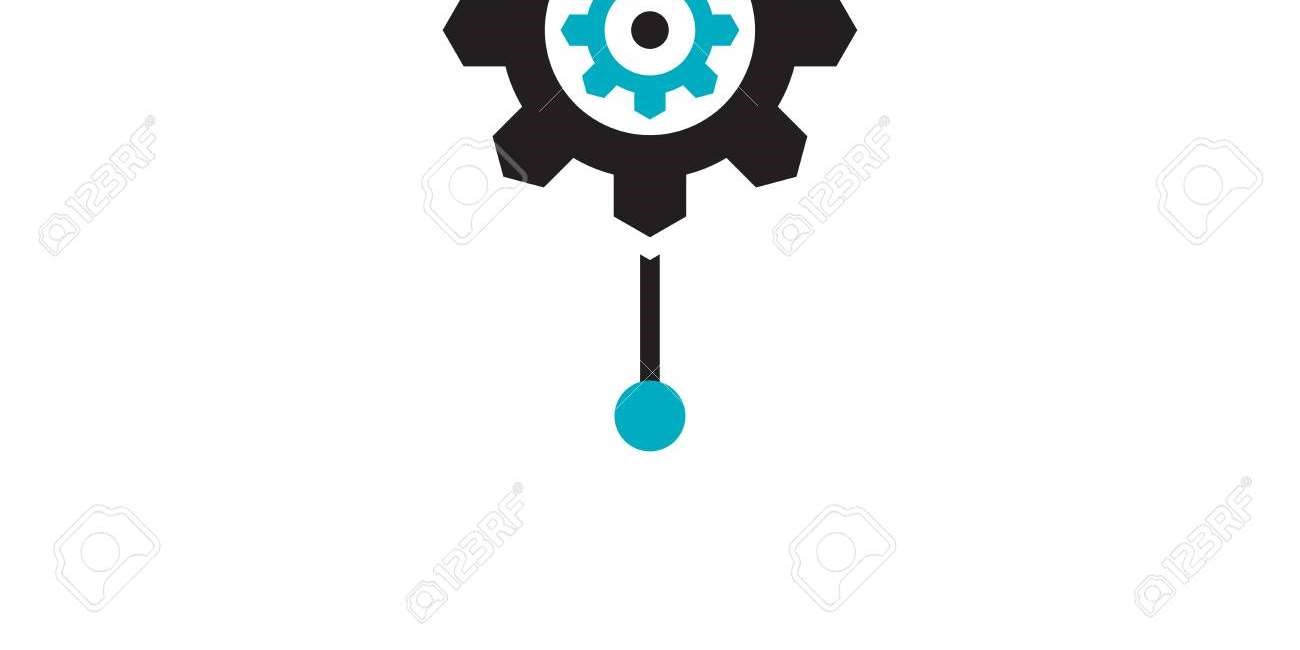
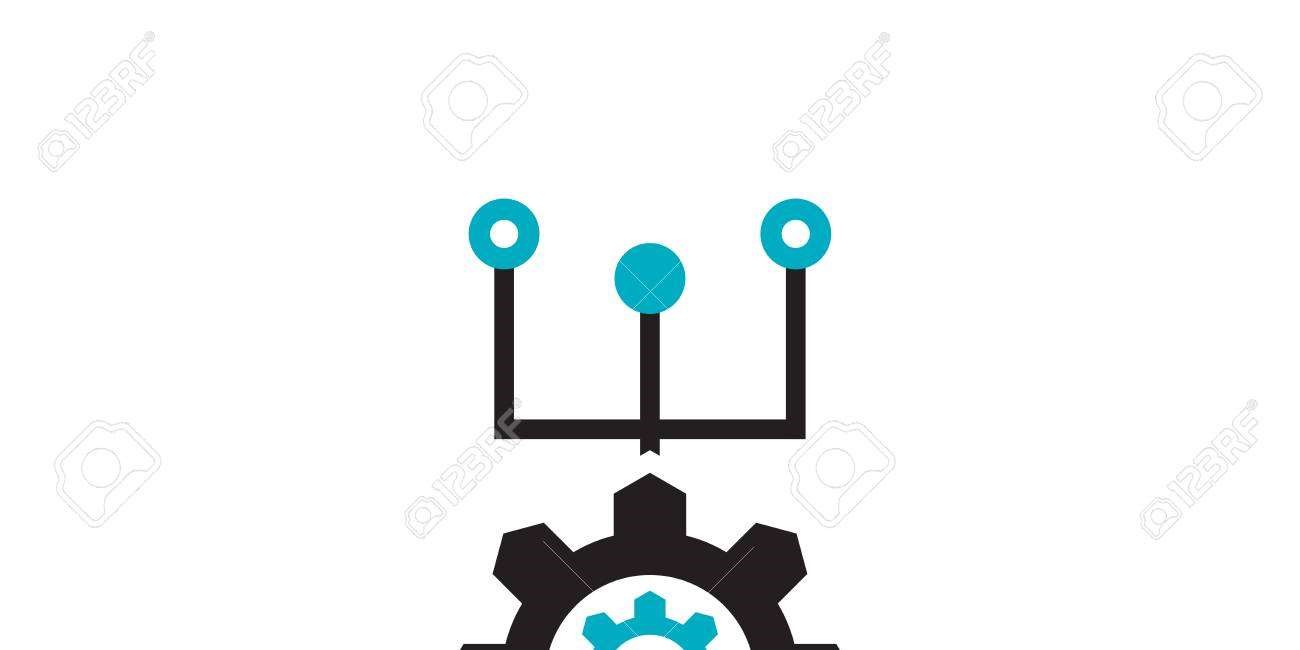
STL



Container



Algorithms



Iterators

Group C: Standard Template Library (STL)



Experiments on STL

1. Title: Experiments onto STL.
2. Objectives: To learn and understand concepts of Standard Template Library
3. Outcomes: To understand the concept of Standard Template Library.
4. Software/Hardware/Other Requirements:
   * Any CPU with Pentium Processor / similar, 256 MB RAM/ more, 1GB HDD / more.
   * Operating System – ubuntu/Fedora 64bit OS  Software: G++ compiler/ GCC compiler, Code Editor
5. Theory:

A. Introduction to STL (Standard Template Library)

* The C++ STL (Standard Template Library) is a powerful set of C++ template classes to provides general-purpose templatized classes and functions that implement many popular and commonly used algorithms and data structures like vectors, lists, queues, and stacks.
* At the core of the C++ Standard Template Library are following three well-structured components:

|  |  |
| --- | --- |
| Component | Description |
| Containers | Containers are used to manage collections of objects of a certain kind. There are several different types of containers like deque, list, vector, map etc. |
| Algorithms | Algorithms act on containers. They provide the means by which you will perform initialization, sorting, searching, and transforming of the contents of containers. |
| Iterators | Iterators are used to step through the elements of collections of objects. These collections may be containers or subsets of containers. |

a) Containers: Standard Containers

* A container is a holder object that stores a collection of other objects (its elements).
* A container is a way to store data, whether the data consists of built-in types such as int and float, or of class objects.
* STL defines 10 containers which are grouped into 3 categories as shown in figure.
* Each container class defines a set of function that can be used to manipulate its contents.

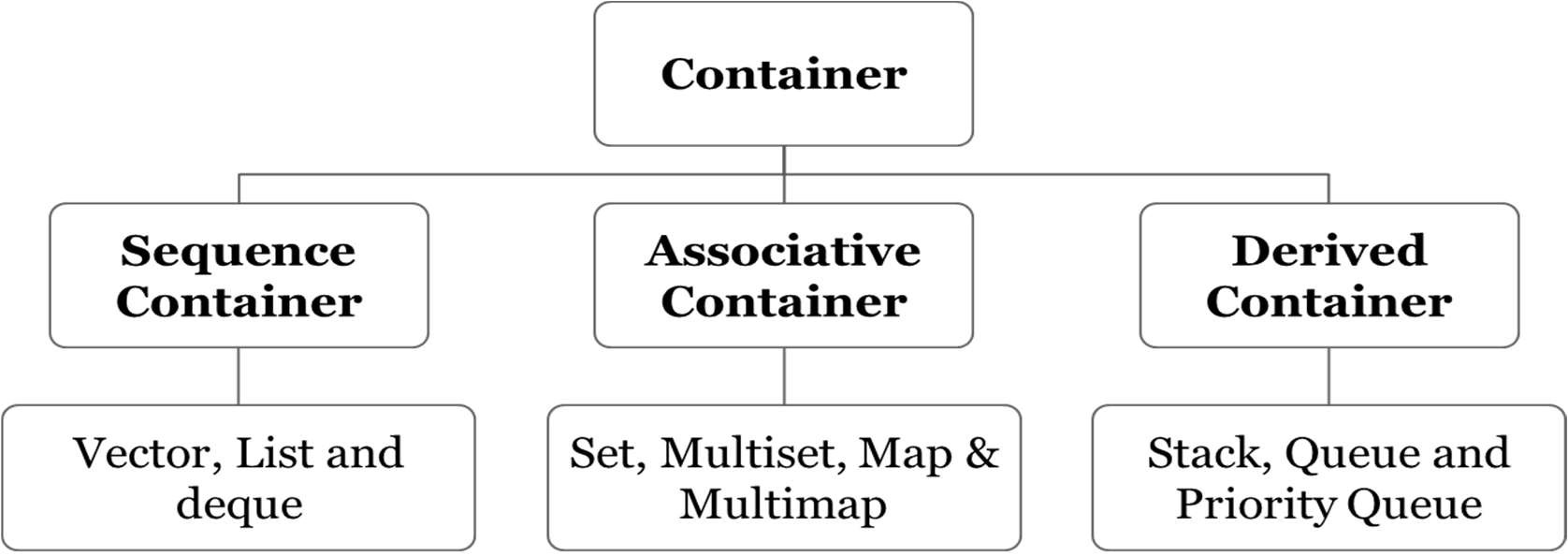
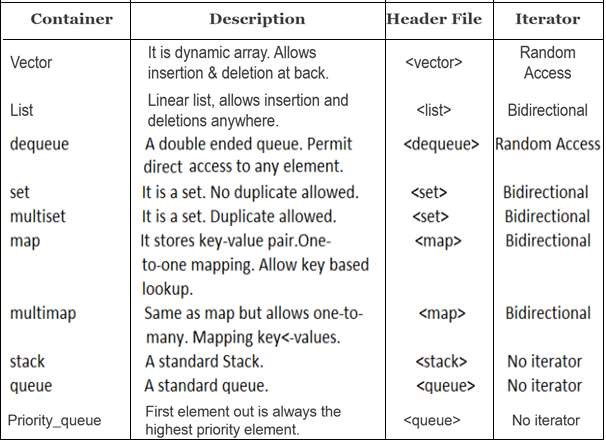


Table: Containers Overview

 b) Algorithm:

* An algorithm is a function that does something to the items in a container (or containers).
* Algorithms in the STL are not member functions or even friends of container classes, as they are in earlier container libraries.
* They are just standalone template functions, which are included in <algorithm> header file. STL algorithms reinforce the philosophy of reusability.
* Various Categories of Algorithms:

1. Sorting Algorithms
2. Mutating Sequence Algorithms
3. Non-Mutating Sequence Algorithms
4. Numerical Algorithms
5. Set Algorithms
6. Relational Algorithms

* Suppose you create an array of type int, with data in it:

int arr[8] = {42, 31, 7, 80, 2, 26, 19, 75};

* You can then use the STL sort() algorithm to sort this array by saying sort(arr, arr+8);
* where arr is the address of the beginning of the array, and arr+8 is the past-the-end address (one item past the end of the array).

c) Iterators

* Iterators are pointer-like entities that are used to access individual data items from a container.
* Often, they are used to move sequentially from element to element, a process called iterating through the container.
* You can increment iterators with the ++ operator so they point to the next element, and dereference them with the \* operator to obtain the value of the element they point to.
* In the STL an iterator is represented by an object of an iterator class.
* There are 3 major classes of iterators: forward, bidirectional, and random access.
* Iterators are 5 types:

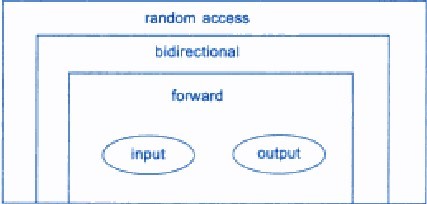
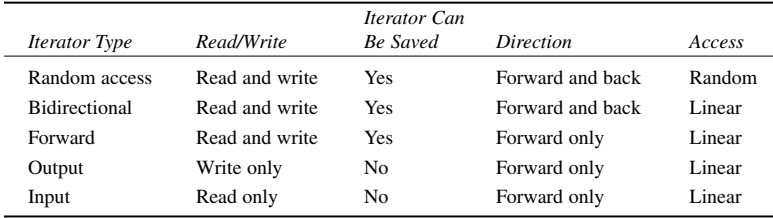
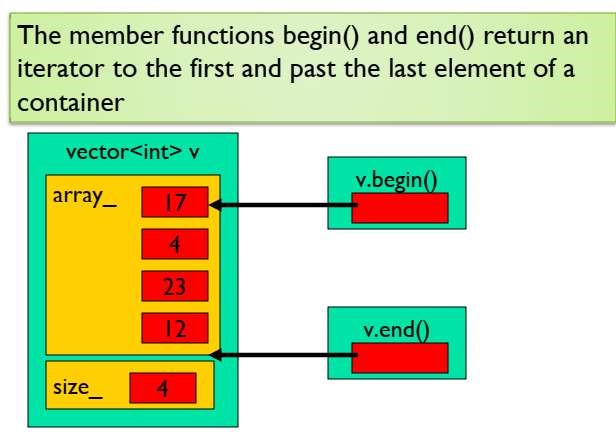
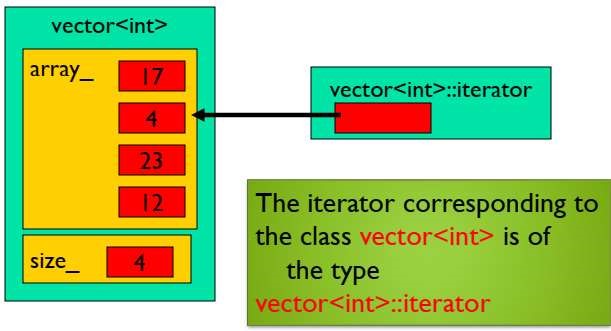


Fig: Functionality Venn diagram of iterators

* Working of Iterator:



Experiment No.7

1. Title: Associative Container: Map.
2. 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 populations 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.

1. Objectives:
   1. To understand the map STL class and its methods.
   2. To understand how to use of STL map class for finding population from the map object.
2. Outcomes:
   1. Students will be able to demonstrate use of STL in C++.
   2. Student will be able to implement the various operations using map class.
3. Software/Hardware/Other Requirements:
   * Any CPU with Pentium Processor / similar, 256 MB RAM/ more, 1GB HDD / more.
   * Operating System – ubuntu/Fedora 64bit OS  Software: G++ compiler/ GCC compiler, Code Editor
4. Theory:

A. Associative Container:

* + They are not sequential; instead it uses keys to access data.
  + Elements in an associative container are stored and retrieved by a key. To access an element in an associate container, we use the key of the element. Associative containers are normally implemented as a binary search tree.
  + The keys, typically numbers or stings, are used automatically by the container to arrange the stored elements in a specific order.
  + It’s like an ordinary English dictionary, in which you access data by looking up words arranged in alphabetical order.
  + If you know the key, you can access the associated value swiftly & easily.
  + There are two kinds of associative containers in the STL: sets and maps.
  + The standard has recently added multiset and multimap with duplicate values and duplicate keys.
  + These both store data in a structure called a tree, which offers fast searching, insertion, and deletion.
  + However, these are very slow for random access and inefficient for sorting.
  + Creating associative containers is just like creating sequential ones: set<int> intSet; //create a set of ints or

typedef map<string,int> phoneMap;

//create map with string as key and int as value.

B. Map

* A map, which is also called a table, a dictionary, or an associate array, is defined in the <map> header file. It is a container that stores a template pair of key and value.
* The elements are sorted in ascending order based on the key. In a map, the keys are unique. Figure shows an example of a map.

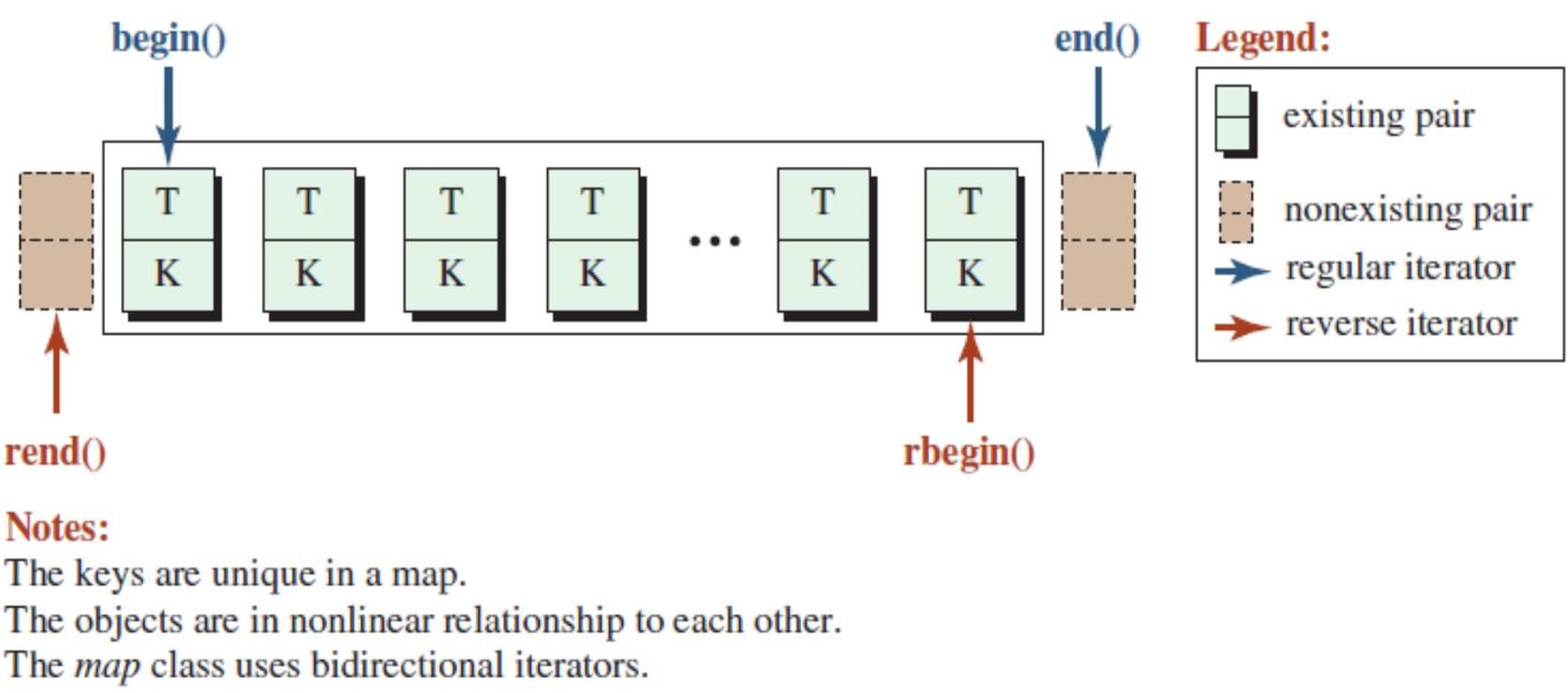


Fig: Map

* Map associative container are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped value. No two mapped values can have same key values. map::operator[]
* This operator is used to reference the element present at position given inside the operator. It is similar to the at() function, the only difference is that the at() function throws an out-of-range exception when the position is not in the bounds of the size of map, while this operator causes undefined behaviour.
* To use a map, we must include the header file <map>
* Syntax: mapname[key] o Parameters: Key value mapped to the element to be fetched. o Returns: Direct reference to the element at the given key value.
* Examples:

Input: map mymap; mymap['a'] = 1; mymap['a']; Output: 1

* The template for the map defined inside namespace std is: namespace std {

template <class Key, class T, class Compare = less<Key>, class Allocator = allocator<pair<const Key, T> > > class map;

* The elements of a map may have any types of key and value that meet the following two requirements: o The key/value pair must be assignable and copyable. o The key must be comparable with the sorting criterion.

C. Built-in functions

* STL Maps come with some basic built-in functions. Some of them are explained below:
  1. begin(): Returns an iterator to the first element in the map.
  2. end(): Returns an iterator pointing to the past-end.
  3. size(): Returns the number of elements in the map.
  4. empty(): Returns a boolean value indicating whether the map is empty.
  5. insert( pair(key, value)): Adds a new key-value pair to the map. An alternate way to insert values in the map is:

map\_name[key] = value;

* 1. find(val): Gives the iterator to the element val, if it is found otherwise it returns m.end()
  2. erase(iterator position): Removes the element at the position pointed by the iterator.
  3. erase(const g): Removes the key value g from the map.
  4. clear(): Removes all the elements from the map.

1. Algorithm:
   1. Start.
   2. Give a header file to map associative container.
   3. Insert states name so that we get values as population of that state.
   4. Use populationMap.insert().
   5. Display the population of states.
   6. End.
2. Implementation:

|  |
| --- |
| C++ Program for implementation Personal Record using vector Container and  Searching, Sorting algorithms |

#include <iostream>

#include <map>

#include<iomanip>

using namespace std;

int main()

{

    typedef map<string, long> mapType;

    mapType populationMap;

string states[]={"Uttar Pradesh","Maharashtra","Bihar","West Bengal","Andhra Pradesh","Madhya Pradesh","Tamil Nadu","Rajasthan","Karnataka","Gujarat","Orissa","Kerala","Jharkhand","Assam","Punjab","Chhattisgarh","Haryana","Uttarakhand","Himachal Pradesh","Tripura","Meghalaya","Manipur","Nagaland","Goa","Arunachal Pradesh","Mizoram","Delhi","Jammu and Kashmir + Ladakh"},state;

long populations[]={199812341,112374333,104099452,91276115,84580777,72626809,72147030,68548437,61095297,60439692,41974218,33406061,32988134,31205576,

27743338,25545198,25351462,10086292,6864602,3673917,2966889,2855794,1978502,1458545,1383727,1097206,16787941,12541302},population;

    for(int i=0;i<28;i++)

        populationMap.insert(pair<string, long>(states[i],populations[i]));

    int cho;

    char ch;

    mapType::iterator iter = populationMap.begin();

    do

    {

        cout<<"\n\t\t Map Menus"<<endl;

        cout<<"\t-----------------------------------------------------"<<"\n\t 1. Display all States Population"<<"\n\t 2. Find Particular State Population"<<"\n\t 3. Update Population"<<"\n\t 4. Size of Population Map"<<"\n\t 5. Add another State Population"<<"\n\t 6. Exit"

<<"\n\t-----------------------------------------------------"<<"\n\t Enter your choice:";

        cin>>cho;

        switch(cho)

        {

            case 1: cout<<"\n\t"<<left<<setw(30)<<"State "<<setw(20)<<"Population"<<endl;

                   cout<<"\t-----------------------------------"<<endl;

                    for (iter = populationMap.begin(); iter != populationMap.end(); ++iter)

                    cout <<"\t"<<left<<setw(30)<<iter->first <<setw(20)<< iter->second <<endl;

                    cout<<"\t------------------------------------"<<endl;

                    break;

            case 2: cout<<"\n\t Enter the State Name:";

                    fflush(stdin);

                    getline(cin,state);

                    iter = populationMap.find(state);

                    if( iter != populationMap.end() )

                     cout <<"\n\t "<<state <<"'s populations is "<< iter->second <<endl;

                    else

                     cout << "\n\t Key is not in populationMap";

                    break;

            case 3: cout<<"\n\t Enter the State Name:";

                    fflush(stdin);

                    getline(cin,state);

                    cout<<"\n\t Enter Updated Population Count:";

                    cin>>population;

                    populationMap[state]=population;

                    break;

            case 4: cout<<"Size of population :"<< populationMap.size();

                    break;

            case 5: cout<<"\n\t Enter the State Name:";

                    fflush(stdin);

                    getline(cin,state);

                    cout<<"\n\t Enter Population Count:";

                    cin>>population;

                    populationMap.insert(pair<string, long>(state,population));

                    break;

            case 6: exit(0);

            default:cout<<"\n\t Enter choice inbetween 1 to 6 only.";

                    break;

        }

        cout<<"\n\t Do you want to continue(y/n):";

        fflush(stdin);

        cin>>ch;

    }while(ch=='y'||ch=='Y');

    return 0;

}

Output:

Map Menus

* + 1. Display all States Population
    2. Find Particular State Population
    3. Update Population
    4. Size of Population Map
    5. Add another State Population
    6. Exit

----------------------------------------------------- Enter your choice:1

State Population

----------------------------------------------------- Andhra Pradesh 84580777

Arunachal Pradesh 1383727

Haryana 25351462

Madhya Pradesh 72626809

Maharashtra 112374333

Punjab 27743338

Rajasthan 68548437

West Bengal 91276115

-----------------------------------------------------

Do you want to continue(y/n):y

----------------------------------------------------- Enter your choice:2

Enter the State Name:Punjab

Punjab's populations is 27743338

Do you want to continue(y/n):y

----------------------------------------------------- Enter your choice:3

Enter the State Name:Maharashtra

Enter Updated Population Count:45236511

Do you want to continue(y/n):y

----------------------------------------------------- Enter your choice:4

Size of populationMap: 28

Do you want to continue(y/n):y

----------------------------------------------------- Enter your choice:5

Enter the State Name:Sikkim

Enter Population Count:5400005

Do you want to continue(y/n):y

----------------------------------------------------- Enter your choice:4

Size of populationMap: 29

Do you want to continue(y/n):y

1. Conclusion:

Hence, we have successfully studied the concept of STL (Standard Template Library) and how it makes many data structures easy. Also, we had walked through what a C++ map is and how to use one. Also seen few examples of adding elements, reading them and traversing the map.

1. Review Questions & Exercises:
   1. Fill in the Blanks
      1. Map is implemented using BST(binary search tree).
      2. (MULTIMAPS) can have same keys whereas the (MAPS) cannot.
      3. By using (ITERATOR) of the elements in the associate container can be efficiently accessed.
      4. Associative Containers that implements sorted data structures for fast search in (O(log N)) time complexity.
   2. Answer the following:
      1. What is an associative container in C++?

**associative containers** refer to a group of class templates in the [standard library](https://en.wikipedia.org/wiki/C%2B%2B_Standard_Library) of the [C++ programming language](https://en.wikipedia.org/wiki/C%2B%2B) that implement ordered [associative arrays](https://en.wikipedia.org/wiki/Associative_arrays).[[1]](https://en.wikipedia.org/wiki/Associative_containers#cite_note-std-1) Being [templates](https://en.wikipedia.org/wiki/Template_(programming)), they can be used to store arbitrary elements, such as integers or custom classes. The following containers are defined in the current revision of the C++ standard: set, map, multiset, multimap. Each of these containers differ only on constraints placed on their elements.

* + 1. How to declare a map?

map< key\_type, value\_type > map\_name;

* + 1. Explain Associative mapping with example?

Associative mapping associates a particular key with its value.

For example

in map<int, string> record{{1, “krishna”}, {2, “anupam”}, {3, “ajay”}}; ,

maps the integer type values with string type values i.e. integer 1 with string “krishna” and integer 2 with string “anupam” and so on.

* 1. Algorithm Assignments:
     1. Assume we have inserted the pairs (15, 10), (16, 10), (6, 16), and (12, 8) into a map named mp. What would be printed from the following code?

cout << mp [6] << endl;

cout << mp [16] << endl; cout << mp [10] << endl;

output:

16

10

0

* + 1. Write the code to create a map and use the insert member function to insert the following pairs in it: (3, 10), (5, 12), and (7, 8).

map<int, int> mp;

mp.insert({3, 10});

mp.insert({5, 12});

mp.insert({7, 8});

* + 1. Assume we insert the following values into a set: 20, 17, 20, 14, 15, 19, 17, and 10. What would be printed from the size () member function after all insertions?

6 as it considers only unique values.

* 1. Programming Assignments:
     1. Write a program in C++ to use map associative container. The keys will be the names of countries and the values will be the capitals of the countries. When the program runs, the user is prompted to type the name of a country. The program then looks in the map, using the country name as an index and returns the capital of the country.

#include<bits/stdc++.h>

using namespace std;

int main()

{

map<string, string> country{{"India", "New Delhi"},

{"Pakistan", "Islamabad"}, {"Nepal", "Kathmandu"},

{"China", "Beijing"},{"Lebanon", "Beirut"}, {"Maldives", "Male"}, {"Mexico", "Mexico City"},{"South Korea", "Seoul"},

{"Vietnam", "Hanoi"},{"USA", "Washington D.C"}, {"UAE", "Abu Dhabi"}};

string c;

char choice;

while(true)

{

cout<<"\n Enter the name of the country :: ";

cin>>c;

if(country[c] != "")

cout<<"\t The capital of "<<c<<" is "<<country[c]<<endl;

else

cout<<"\t Sorry!! The data for this country is not available \n";

cout<<"\n Want to continue?(Y/N) ";

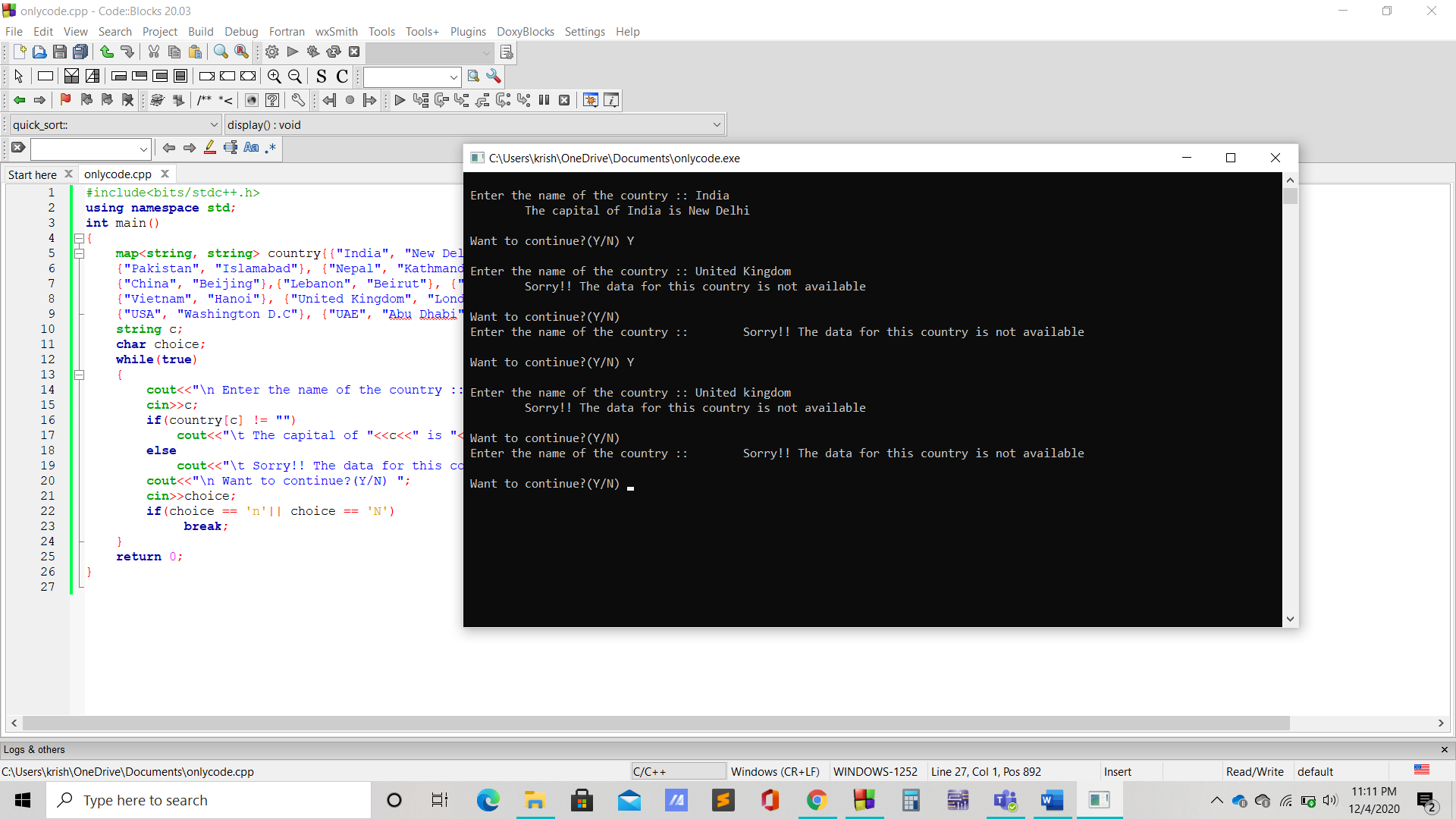
cin>>choice;

if(choice == 'n'|| choice == 'N')

break;

}

return 0;

}

Output:

* + 1. Write a program in C++ to use set associative container for storing 2 set of values. And apply the various set operations on both set containers.

#include<bits/stdc++.h>

using namespace std;

int main()

{

set<int> setA{10,45,65,87,64,23,34};

set<int> setB{97,84,81,65,3,5,6,23,45};

set<int> union\_, intersection, DiffA , DiffB;

cout<<"\n ------Elements of Set A ----\n\t ";

for(auto itr = setA.begin(); itr != setA.end(); itr++)

{

cout<<\*itr<<"\t";

union\_.insert(\*itr);

if(setB.find(\*itr) != setB.end())

{

intersection.insert(\*itr);

}

else

{

DiffA.insert(\*itr);

}

}

cout<<"\n------- Elements of set B -------- \n\t ";

for(auto itr = setB.begin(); itr != setB.end(); itr++)

{

cout<<\*itr<<"\t";

union\_.insert(\*itr);

if(setA.find(\*itr) == setA.end())

{

DiffB.insert(\*itr);

}

}

cout<<"\n--------------------Union of two sets ------------------ \n\t ";

for(auto itr = union\_.begin(); itr != union\_.end(); itr++)

cout<<\*itr<<"\t";

cout<<"\n\n----------------------- Intersection of two sets --------------------------\n\t ";

for(auto itr = intersection.begin(); itr != intersection.end(); itr++)

cout<<\*itr<<"\t";

cout<<"---------------------------- Elements present in only set A(Symmetric Difference of A) ---------------------- \n\t ";

for(auto itr = DiffA.begin(); itr != DiffA.end(); itr++)

cout<<\*itr<<"\t";

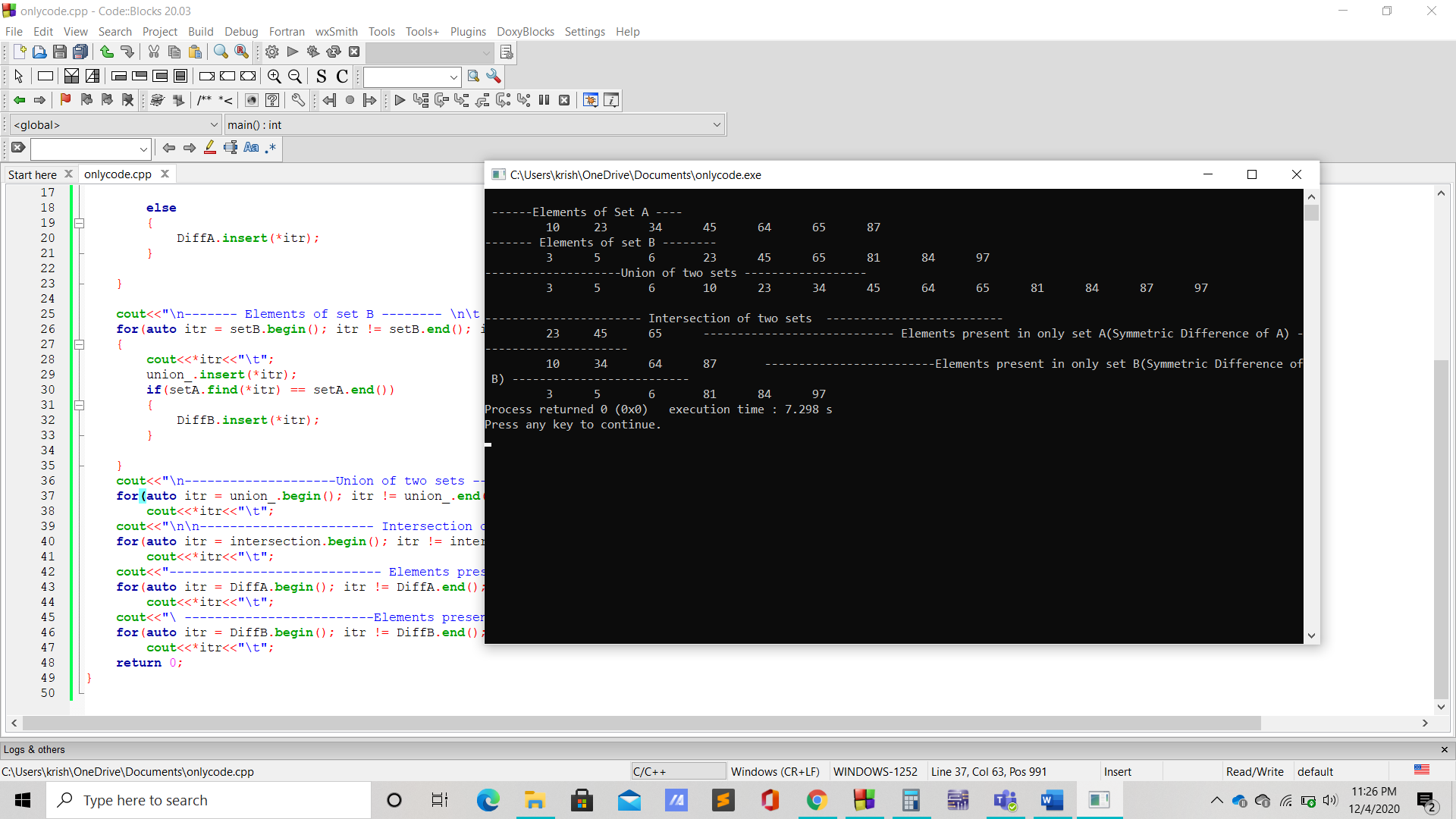
cout<<"\ -------------------------Elements present in only set B(Symmetric Difference of B) -------------------------- \n\t ";

for(auto itr = DiffB.begin(); itr != DiffB.end(); itr++)

cout<<\*itr<<"\t";

return 0;

}

Output:

1. References:
   1. E Balagurusamy Object-Oriented Programming with C++.7th edition. McGraw-

Hill Publication, ISBN 10: 9352607996 ISBN 13: 9789352607990

* 1. Behrouz A. Forouzan, Richard F. Gilberg C++ Programming: An Object-Oriented

Approach Published by McGraw-Hill Education, ISBN 978-0-07-352338-5

* 1. Tony Gaddis- “STARTING OUT WITH C++ From Control Structures through Objects”, Pearson Education, ISBN 13: 978-0-13-376939-5