**Container**

**STL Algorithms**

**Iterators**



**Group C: Standard Template Library (STL)**

1. **| P a g e A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**

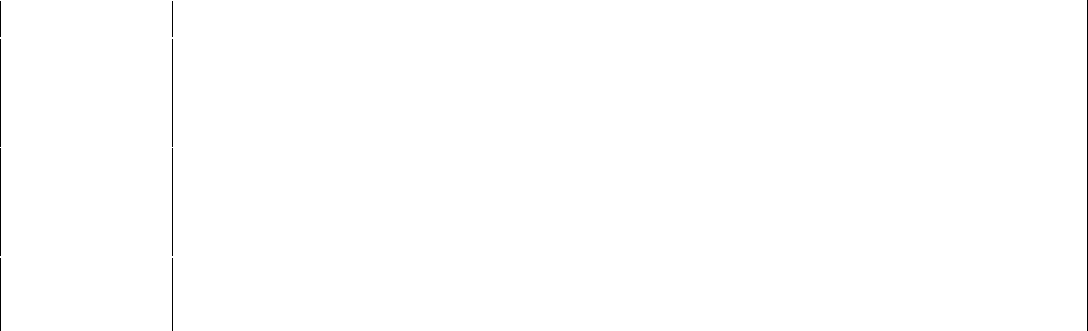
# Experiments on STL

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**

1. **Theory:**
   1. **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:**

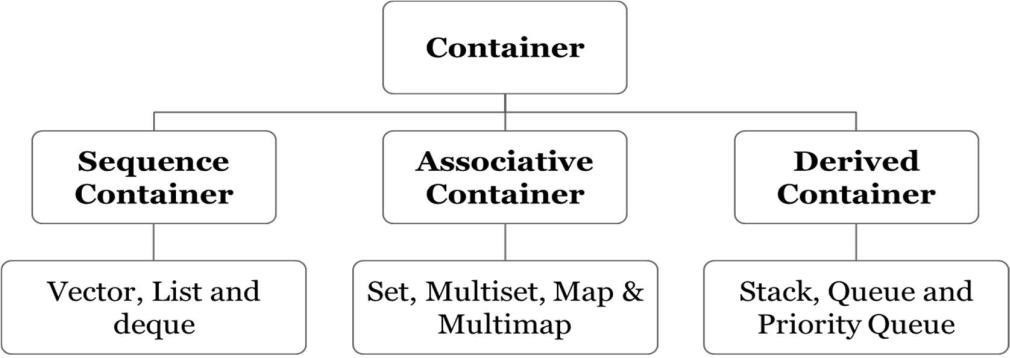
|  |  |
| --- | --- |
| **Component** | **Description** |
| **Containers** | **Containers are used to manage collections of objects of a certain kind.** |
|  | **There are several different types of containers like deque, list, vector,** |
|  | **map etc.** |
| **Algorithms** | **Algorithms act on containers. 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.** |

* + 1. **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**

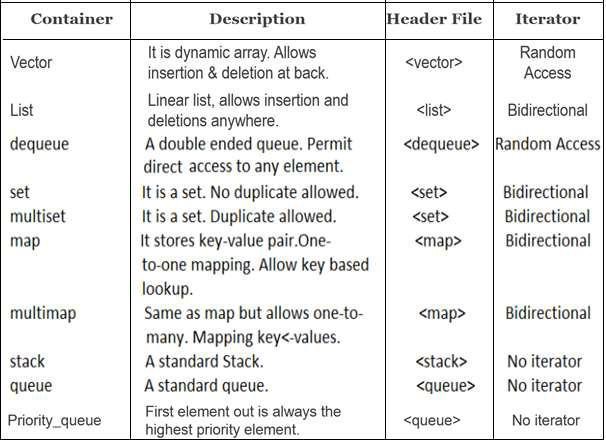
**as int and float, or of class objects.**

**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.**

1. **| P a g e A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**

**Table: Containers Overview**



* + 1. **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).**

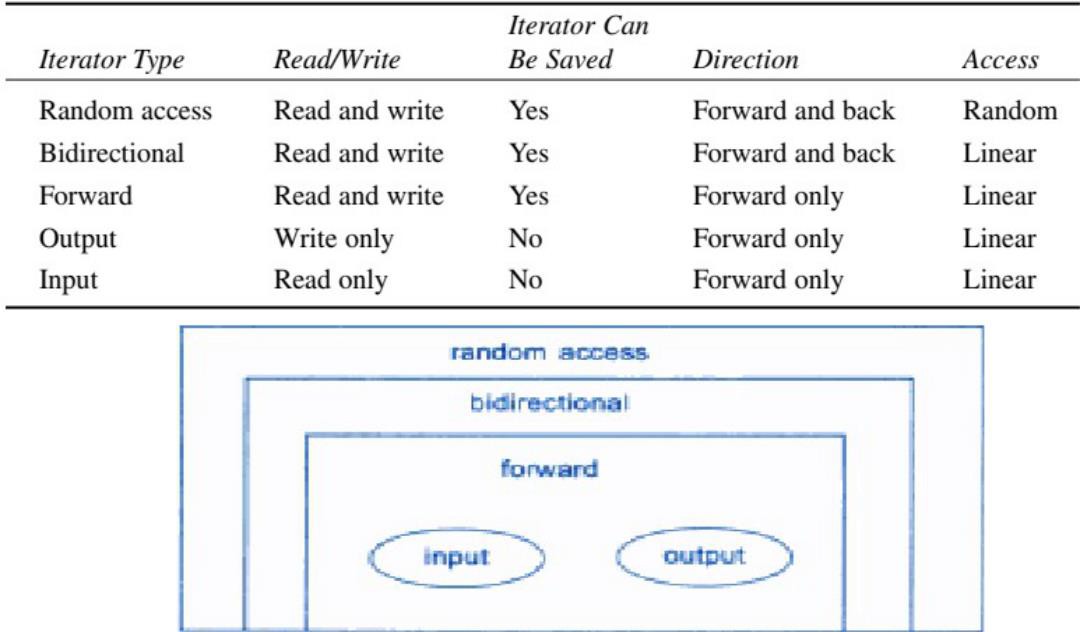
1. **| P a g e A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**
   * 1. **Iterators**

**Iterators are pointer-like entities that are used to access individual data items from a container.**

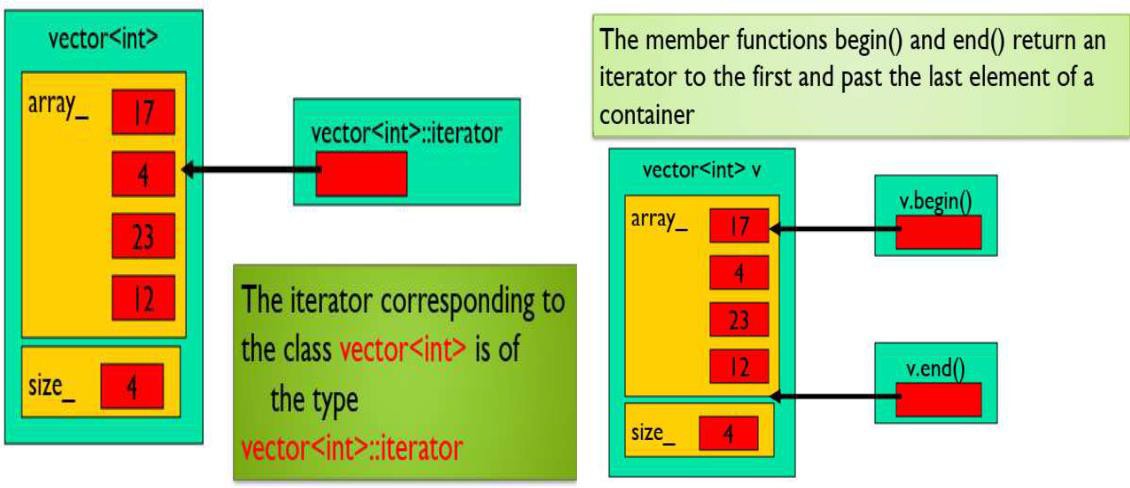
**Often, they are used to move sequentially from element to element, a process called iterating through the container.**

**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:**

1. **| P a g e A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**

# Experiment No.6

1. **Title: User defined datatype for vector STL class.**
2. **Problem Statement:**
   1. **Write C++ program using STL for sorting and searching user defined records such as personal records (Name, DOB, Telephone number etc.) using vector container.**
   2. **Write C++ program using STL for sorting and searching user defined records such as Item records (Item code, name, cost, quantity etc) using vector container.**
3. **Objectives:**
4. **To understand the Vector STL class and its methods.**
5. **To understand how to use of STL vector class for sorting & searching from user defined datatype.**
6. **Outcomes:**
7. **Students will be able to demonstrate use of STL in C++.**
8. **Student will be able to implement the sorting & Searching operations using vector.**
9. **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**

1. **Theory:**
2. **Sequence Container:**

**Stores a set of elements in what you can visualize as a line, like houses on a street. Each element is related to the other elements by its position along the line. Each**

**element (except at the ends) is preceded by one specific element and followed by another.**

**They all expand themselves to allow insertion of elements and all of them support number operations on them.**

**Elements in all these containers can be accessed using an iterator.**

**Fig: Sequence Container**

**Instantiating an STL container object is easy. First you must include an appropriate header file.**

**Then you use the template format with the kind of objects to be stored as the parameter.**

**Examples might be**

**vector<int> aVect; //create a vector of ints list<airtime> departure\_list; //create a list of airtimes**

**Notice that there’s no need to specify the size of STL containers. The containers themselves take care of all memory allocation.**

1. **| P a g e A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**
2. **Vector**

**Vectors are sequence containers that store elements. Deals with dynamic data.**

**C++ vectors may expand depending on the elements they contain. That makes it different from a fixed-size array.**

**C++ vectors can automatically manage storage. It is efficient if you add and delete data often.**

**How to create C++ vector?**

**o Syntax: vector <type> variable (elements) o e.g. vector <int> rooms (9);**

**type defines a data type stored in a vector (e.g., <int>, <double> or <string>).**

**variable is a name that you choose for the data. elements specified the number of elements for the data.**

**It is mandatory to determine the type and variable name. However, the number of elements is optional.**

**All the data elements are stored in contiguous storage. Whenever you want to access or move through the data, you can use iterators.**

**The data elements in vectors are inserted at the end. Use modifiers to insert new elements or delete existing ones.**

**Vector class functions: o Iterators**

* 1. **begin() – Returns an iterator pointing to the first element in the vector**
  2. **end() – Returns an iterator pointing to the theoretical element that follows the last element in the vector**
  3. **rbegin() – Returns a reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element**
  4. **rend() – Returns a reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)**
* **Capacity**

1. **size () – Returns the number of elements in the vector.**
2. **capacity() – Returns the size of the storage space currently allocated to the vector expressed as number of elements.**
3. **resize(n) – Resizes the container so that it contains ‘n’ elements.**
4. **empty() – Returns whether the container is empty.**
5. **reserve() – Requests that the vector capacity be at least enough to contain n elements.**

* **Modifiers:**

1. **push\_back() – It push the elements into a vector from the back**
2. **pop\_back() – It is used to pop or remove elements from a vector from the back.**
3. **insert() – It inserts new elements before the element at the specified position**
4. **erase() – It is used to remove elements from a container from the specified position or range.**
5. **clear() – It is used to remove all the elements of the vector container**
6. **| P a g e A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**
7. **Iterators**

**An iterator allows you to access the data elements stored within the vector. Iterator is an object that functions as a pointer.**

**C++ vectors support random access iterators.**

1. **The for\_each algorithm:**

**Apart from the generic looping techniques, such as “for, while and do- while”, C++ in its language also allows us to use another functionality which solves the same purpose termed “for-each” loops.**

**This loop accepts a function which executes over each of the container elements. This loop is defined in the header file “algorithm”, and hence has to be included for successful operation of this loop.**

**Why to use for\_each?**

**o It is versatile, i.e can work with any container. It reduces chances of errors one can commit using generic for loop. It makes code more readable. for\_each loops improve overall performance of code**

**Syntax:**

**for\_each (InputIterator first, InputIterator last, Function fn)**

* **strt\_iter : The beginning position from where function operations has to be executed.**
* **last\_iter : This ending position till where function has to be executed.**
* **fnc/obj\_fnc : The 3rd argument is a function or an object function which operation would be applied to each element.**

**e.g. for\_each(j.begin(), j.end(), printfunction);**

1. **Sort algorithm**

**Sorting is one of the most basic functions applied to data. It means arranging the data in a particular fashion, which can be increasing or decreasing.**

**There is a built-in function in C++ STL by the name of sort().**

**This function internally uses IntroSort. In more details it is implemented using hybrid of Quick Sort, Heap Sort and Insertion Sort. By default, it uses QuickSort but if QuickSort is doing unfair partitioning and taking more than N\*logN time, it switches to HeapSort and when the array size becomes really small, it switches to InsertionSort.**

**The prototype for sort is: sort(startaddress, endaddress)**

* **startaddress: the address of the first element of the array**
* **endaddress: the address of the next contiguous location of the last element of the array.**

**sort() takes a third parameter that is used to specify the order in which elements are to be sorted.**

**We can also write our own comparator function and pass it as a third parameter. This “comparator” function returns a value; convertible to bool, which basically tells us whether the passed “first” argument should be placed before the passed “second” argument or not.**

1. **| P a g e A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**
2. **binary\_search() algorithm**

**Binary search is a widely used searching algorithm that requires the array to be sorted before search is applied. The main idea behind this algorithm is to keep dividing the array in half (divide and conquer) until the element is found, or all the elements are exhausted.**

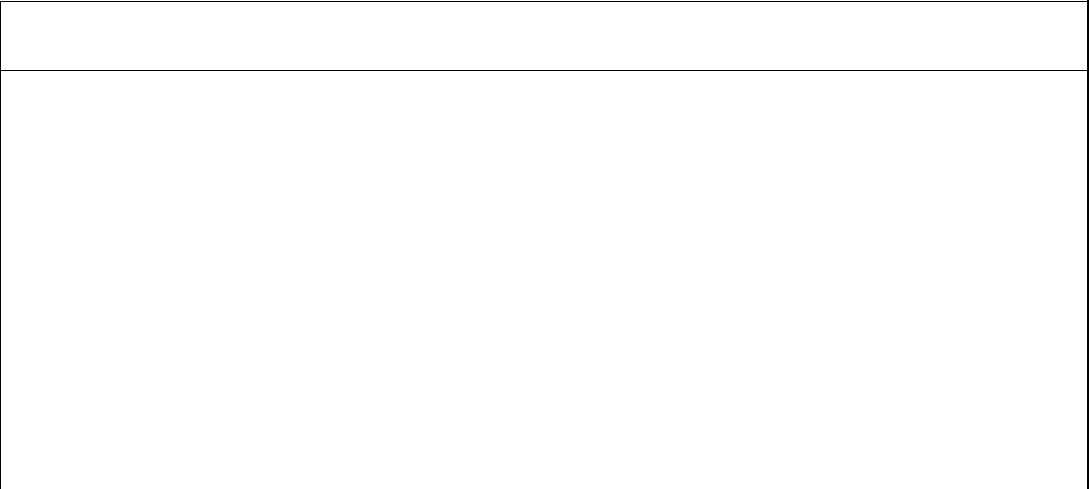
**It works by comparing the middle item of the array with our target, if it matches, it returns true otherwise if the middle term is greater than the target, the search is performed in the left sub-array.**

**If the middle term is less than the target, the search is performed in the right sub-array.**

**The prototype for binary search is: binary\_search(startaddress, endaddress, valuetofind)**

* **startaddress: the address of the first element of the array.**
* **endaddress: the address of the next contiguous location of the last element of the array.**
* **valuetofind: the target value which we have to search for.**

1. **Algorithm:**
2. Start.
3. Give a header file to use 'vector'.
4. Create a vector naming 'personal records'.
5. Initialize variables to store name, birth date and telephone number.
6. Using iterator store as many records you want to store using predefined functions as push\_back().
7. Create another vector 'item record'
8. Initialize variables to store item code, item name, quantity and cost.
9. Using iterator and predefined functions store the data.
10. Using predefined function sort(), sort the data stored according to user requirements.
11. Using predefined function search, search the element from the vector the user wants to check.
12. Display and call the functions using a menu.
13. End.
14. **Implementation:**



**C++ Program for implementation Personal Record using vector Container and**

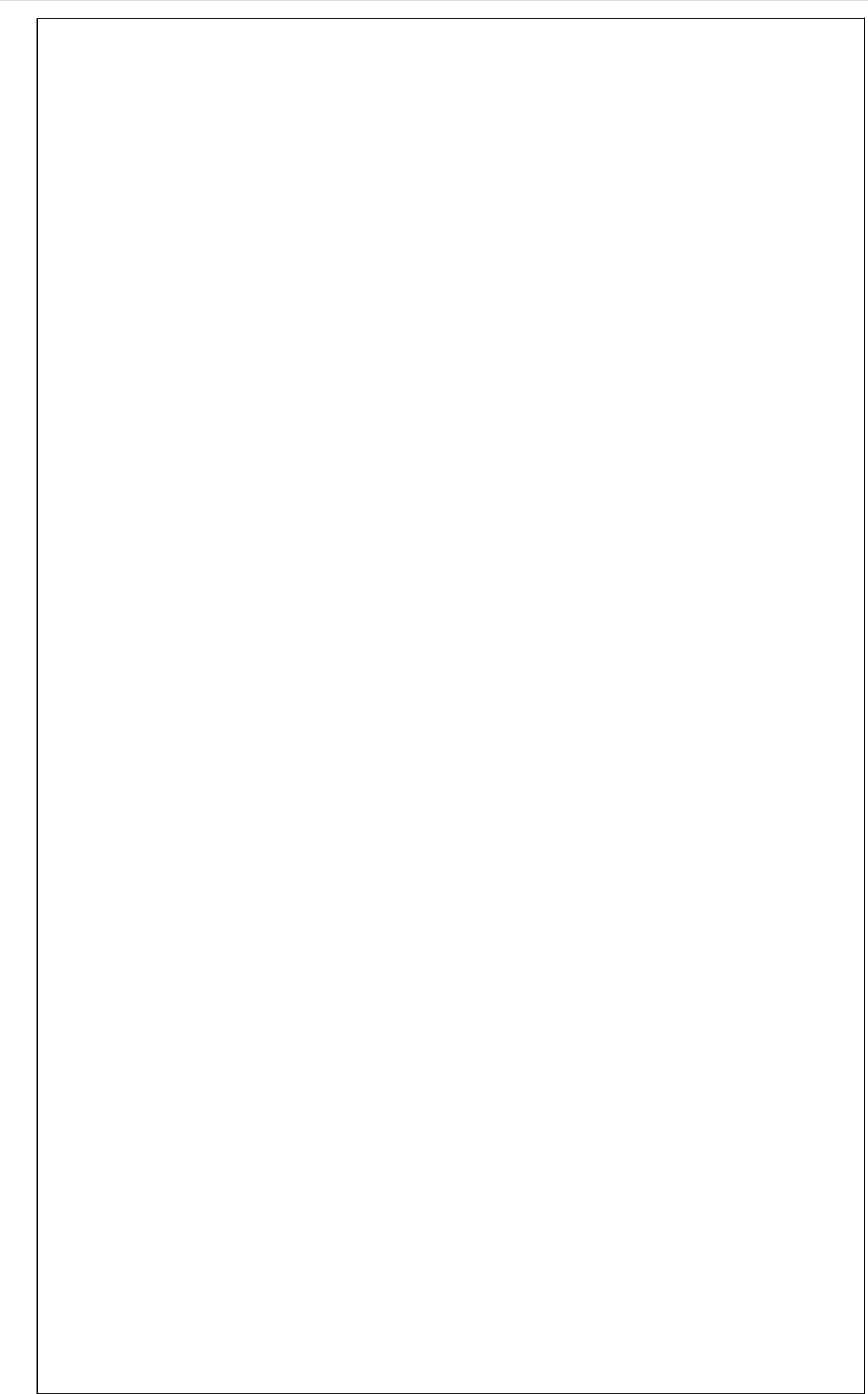
**Searching, Sorting algorithms**

**#include<iostream> #include<vector> #include<algorithm> using namespace std; class Personal{**

**string name,dob,tel\_no; public:**

**Personal(string name1,string dob1,string tel):name(name1),dob(dob1),tel\_no(tel){ }**

|  |  |
| --- | --- |
| **string getName(){ return name;** | **}** |
| **string getDob(){ return dob; }** |  |
| **string getTelNo(){return tel\_no;}** |  |
| **};** | |
| **54 | P a g e** | **A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e** |



**//comparator for sorting & Binary Search Algorithms bool comparator(Personal lhs,Personal rhs){**

**return(lhs.getName() < rhs.getName());**

**}**

**void print(Personal& p)**

**{**

**cout<<"\n\t"<<p.getName()<<"\t"<<p.getDob()<<"\t"<<p.getTelNo()<<endl;**

**}**

**int main()**

**{**

**vector<Personal> pvec; int cho;**

**char ch;**

**string name,dob,telno; do**

**{**

**cout<<"\n\t Personal Record Menus"<<endl; cout<<"\n ";**

**cout<<"\n\t 1. Create New Record"; cout<<"\n\t 2. Display All Records"; cout<<"\n\t 3. Search Perticular Record"; cout<<"\n\t 4. Sort All Records & display"; cout<<"\n\t 5. Exit";**

**cout<<"\n ";**

**cout<<"\n\t Enter your choice:"; cin>>cho;**

**switch(cho)**

**{**

**case 1: cout<<"\n\t Enter Personal Details:"; cout<<"\n\t Name:";**

**cin>>name;**

**cout<<"\n\t Date of Birth:"; cin>>dob;**

**cout<<"\n\t Telephone Number(10 Digit only):"; cin>>telno; pvec.push\_back(Personal(name,dob,telno)); break;**

**case 2: cout<<"\n\t Personal Records "<<endl; cout<<"\n ";**

**cout<<"\n\tName\tDateofBirth\tTelephoneNo"; cout<<"\n ";**

**for\_each(pvec.begin(),pvec.end(),print); break;**

**case 3: cout<<"\n\t Enter Person name for Searching:"; cin>>name;**

**if (binary\_search(pvec.begin(),pvec.end(),Personal(name,"**

**"," "), comparator))**

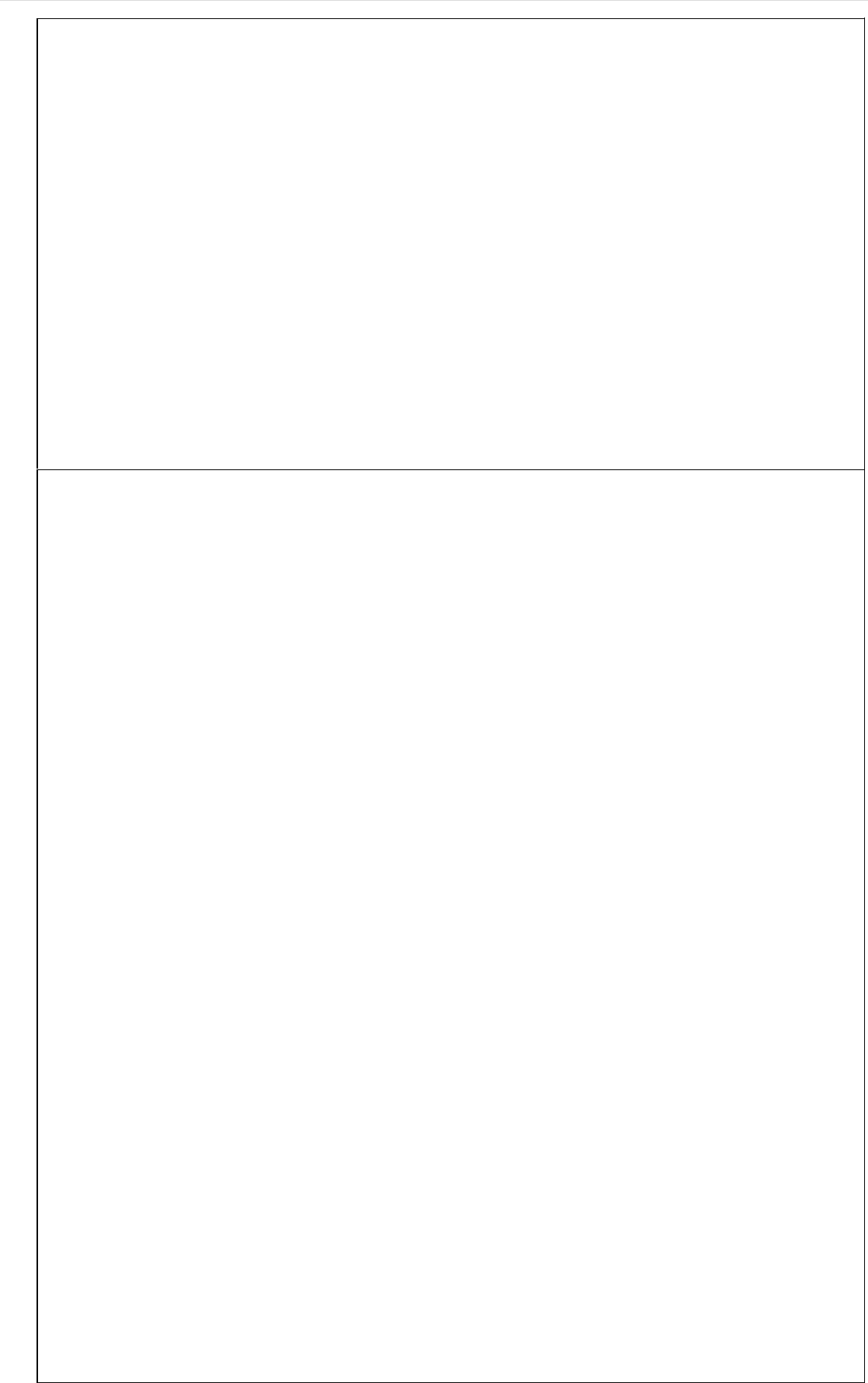
**cout <<"\n\t Person with name "<<name<<" exists in vector.\n";**

**else**

**cout <<"\n\t Person with name "<<name<<" doesnot exists in vector.\n";**

**break;**

1. **| P a g e A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**

**case 4: sort(pvec.begin(),pvec.end(),comparator); cout<<"\n\t Personal Records "<<endl; cout<<"\n ";**

**cout<<"\n\tName\tDateofBirth\tTelephoneNo"; cout<<"\n ";**

**for\_each(pvec.begin(),pvec.end(),print); break;**

**case 5: exit(0);**

**default: cout<<"\n\t Enter choice inbetween 1 to 5 only."; break;**

**}**

**cout<<"\n\t Do you want to Continue(Y/N):"; fflush(stdin);**

**cin>>ch;**

**}while(ch=='Y'||ch=='y'); return 0;**

**}**

**Output:**

**Personal Record Menus**

**----------------------------------------------------**

* 1. **Create New Record**
  2. **Display All Records**
  3. **Search Perticular Record**
  4. **Sort All Records & display**
  5. **Exit**

**----------------------------------------------------**

**Enter your choice:1 Enter Personal Details: Name:Kuldeep**

**Date of Birth:21/07/1989**

**Telephone Number(10 Digit only):8668277166 Do you want to Continue(Y/N):y**

**----------------------------------------------------**

**Enter your choice:1 Enter Personal Details: Name:Aman**

**Date of Birth:25/12/1995**

**Telephone Number(10 Digit only):8421869545 Do you want to Continue(Y/N):y**

**----------------------------------------------------**

**Enter your choice:1 Enter Personal Details: Name:Dilipkumar Date of Birth:05/05/1995**

**Telephone Number(10 Digit only):9421967989 Do you want to Continue(Y/N):y**

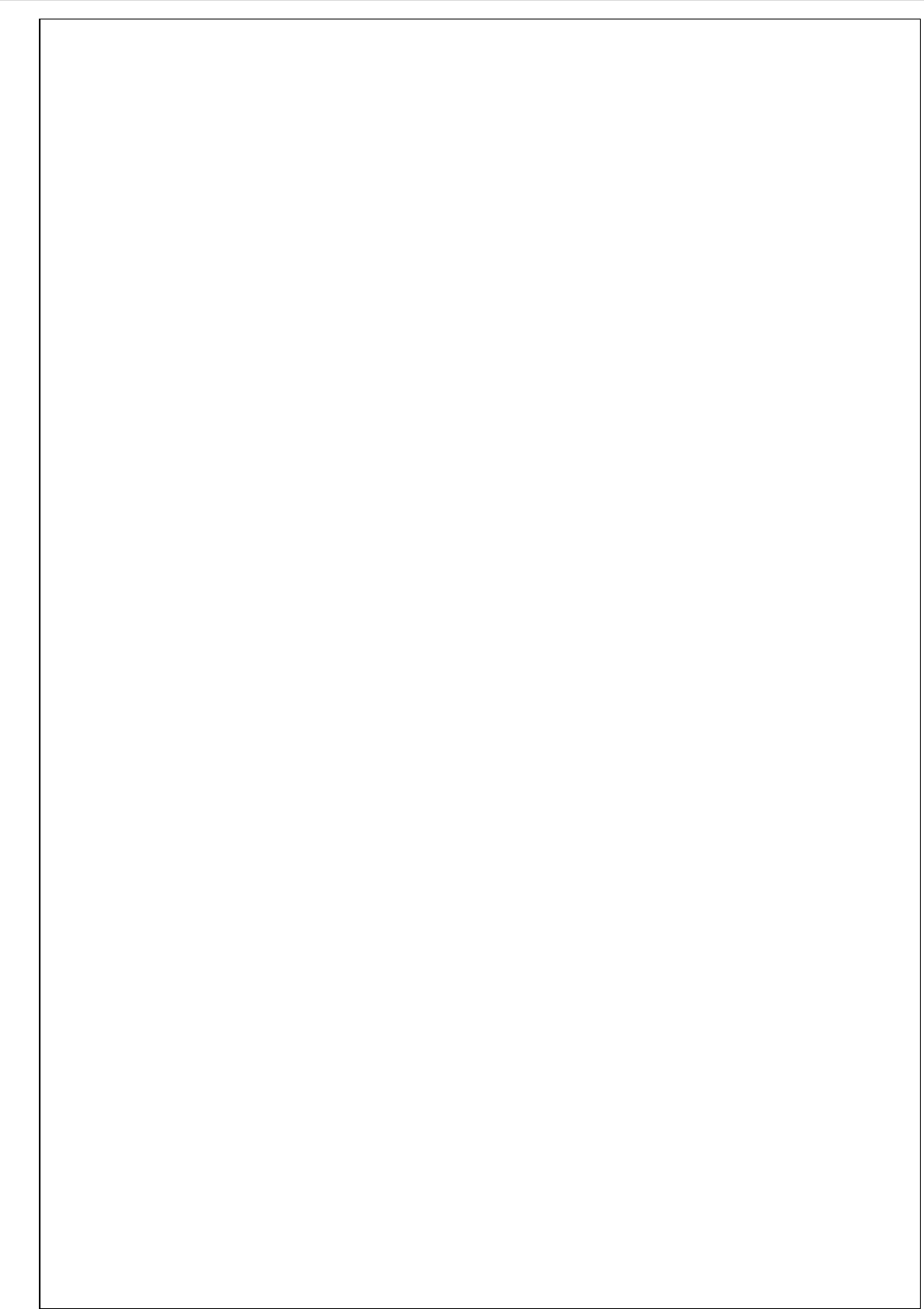
**----------------------------------------------------**

**Enter your choice:1 Enter Personal Details: Name:Sushant**

**Date of Birth:12/12/1997**

**Telephone Number(10 Digit only):8525636521 Do you want to Continue(Y/N):y**

1. **| P a g e A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**



**----------------------------------------------------**

**Enter your choice:2 Personal Records**

**----------------------------------------------------**

**Name DateofBirth TelephoneNo**

**----------------------------------------------------**

|  |  |  |
| --- | --- | --- |
| **Kuldeep 21/07/1989** | | **8668277166** |
| **Aman** | **25/12/1995** | **8421869545** |
| **Dilip** | **05/05/1995** | **9421967989** |
| **Sushant 12/12/1997** | | **8525636521** |

**Do you want to Continue(Y/N):y**

**----------------------------------------------------**

**Enter your choice:3**

**Enter Person name for Searching:Dilipkumar Person with name Dilipkumar exists in vector. Do you want to Continue(Y/N):y**

**----------------------------------------------------**

**Enter your choice:4 Personal Records**

**----------------------------------------------------**

**Name DateofBirth TelephoneNo**

**----------------------------------------------------**

|  |  |  |
| --- | --- | --- |
| **Aman** | **25/12/1995** | **8421869545** |
| **Dilip** | **05/05/1995** | **9421967989** |
| **Kuldeep 21/07/1989** | | **8668277166** |
| **Sushant 12/12/1997** | | **8525636521** |

**Do you want to Continue(Y/N):y**

**----------------------------------------------------**

**Enter your choice:1 Enter Personal Details:**

**Name:Jayesh**

**Date of Birth:14/07/1991**

**Telephone Number(10 Digit only):8668277661 Do you want to Continue(Y/N):y**

**----------------------------------------------------**

**Enter your choice:2 Personal Records**

**----------------------------------------------------**

**Name DateofBirth TelephoneNo**

**----------------------------------------------------**

|  |  |  |
| --- | --- | --- |
| **Aman** | **25/12/1995** | **8421869545** |
| **Dilip** | **05/05/1995** | **9421967989** |
| **Kuldeep 21/07/1989** | | **8668277166** |
| **Sushant 12/12/1997** | | **8525636521** |
| **Jayesh 14/07/1991** | | **8668277661** |

**Do you want to Continue(Y/N):y**

**----------------------------------------------------**

**Enter your choice:5**

1. **Conclusion:**

Hence, we have successfully studied the concept of STL (Standard Template Library) and how it makes many data structures easy. It briefs about the predefined functions of STL and their uses such a search () and sort ().

1. **| P a g e A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**

## Review Questions & Exercises:

* 1. **Fill in the Blanks**
     1. A(n) **sequence** container organizes data in a sequential fashion similar to an array.
     2. A(n) **associative** container uses keys to rapidly access elements.
     3. **Iterators** are pointer-like objects used to access data stored in a container.
     4. STL algorithms are implemented as **functions**.
     5. A collection of generic class and function is called as **Standard Template Library (STL).**
     6. A **Sequence Container** is a collection of objects in which the programmer controls the order of storing and retrieving elements.

## Answer the following:

* + 1. What STL algorithm randomly shuffles the elements in a container? Ans:

random\_shuffle () randomly shuffles the elements in a container. We pass starting and ending iterator as parameters

E.g.: random\_shuffle (myvector.being(), myvector.end()) ;

* + 1. What are the various sorting & searching algorithms available in algorithm?

Ans :

 Some of the popular Sorting algorithms are

* + - * Selection Sort o Bubble Sort o Insertion Sort o Merge Sort o Heap Sort o Radix Sort
      * Introsort – Builtin C++ STL uses this sorting algorithm.

 Some of the popular Searching algorithms are

* + - * Linear Search
      * Binary Search – C++ STL has built in binary\_search() algorithm.
      * Jump Search o Interpolation Search o Exponential Search o Fibonacci Search
    1. What are the various applications of vector container?

Ans: Vectors in C++ are the dynamic arrays that are used to store data. Unlike arrays, which are used to store sequential data and are static in

* nature, Vectors provide more flexibility to the program.

## Algorithm Assignments:

* + 1. A program has the following definition statements: vector<int> numbers;

vector<int>::iterator iter;

Write code that uses the iterator to display all the values stored in the vector.

Ans:

#include<iostream> #include<vector> #include<iterator> using namespace std; int main(){

vector<int> numbers = {14, 15, 16, 17, 13, 20};

vector<int>::iterator iter;

for(iter = numbers.begin(); iter != numbers.end(); iter++) cout<<\*iter<<" ";

}

* + 1. A program has the following definition: vector<double> numbers;

Write code that uses the STL for\_each algorithm to display the

elements of the numbers vector using the display function.

## Ans:

#include<iostream> #include<vector> #include<algorithm> using namespace std; void display(double x){

cout << x << " ";

}

int main(){

vector<double> numbers = {14.1, 15.2, 16.3, 17.4, 13.5,

0.6};

for\_each(numbers.begin(), numbers.end(), display);

}

* + 1. The push\_front operation is not defined for a vector. Write code to insert a value at the front of a vector of type integer.

## Ans:

#include<iostream> #include<vector> #include<algorithm> using namespace std;

void display(vector <int> &v){ for(auto i : v)

cout<<i<<" "; cout<<"\n";

}

void Push\_front(vector <int> &v, int x){ v.insert(v.begin(), x);

}

int main(){

vector<int> numbers = {14};

display(numbers);

Push\_front(numbers, 15);

cout<<"After pushing 15 at front : "; display(numbers);

}

* + 1. The pop\_front operation is not defined for a vector. Write code to remove the first element from a vector of type integer.

Ans:

#include<iostream> #include<vector> #include<algorithm> using namespace std;

void display(vector <int> &v){ for(auto i : v)

cout<<i<<" "; cout<<"\n";

}

void Pop\_front(vector <int> &v){ v.erase(v.begin());

}

int main(){

vector<int> numbers = {15, 14};

|  |  |
| --- | --- |
| display(numbers); |  |
| Pop\_front(numbers); | |

cout<<"After poping from front: "; display(numbers);

}

## Programming Assignments:

* + 1. Write a program that creates a vector of a few names. Then print the names, sort the names, and print the sorted names.

Ans:

#include <bits/stdc++.h> using namespace std;

void display(vector <string> &v){ for(auto i : v)

cout<<i<<" "; cout<<"\n";

}

int main(){

int n;

cout<<"Enter number of names to be added: "; cin>>n;

vector<string> names;

cout<<"Enter names: \n";

for(int i=0;i<n;i++){

string name;

cin>>name;

names.push\_back(name);

}

cout<<"Before sorting the vector names are:\n";

display(names);

sort(names.begin(),names.end());

rting the vector names :\n";

return 0;

cout<<"After so

display(names);

}

* + 1. Write a program using STL vector for Sorting and searching with user- defined records: item record (item code, item name, quantity and cost) Ans:

#include <iostream> #include <algorithm> #include <vector> using namespace std; class Item

{

public:

char name[10];

int quantity;

int cost;

int code;

bool operator==(const Item& i1){

|  |  |  |  |
| --- | --- | --- | --- |
| if(code==i1.code) | | | |
| return true; | | | |
| return false; | | |  |
| } | |  | |
|  |  | | |

bool operator<(const Item& i1){

if(code<i1.code)

return true;

return false;

}

};

vector<Item> o1; void insert(){

Item cur;

cout<<"\nEnter Item Code:"; cin>>cur.code;

cout<<"\nEnter Item Name:"; cin>>cur.name;

cout<<"\nEnter Item Quantity:"; cin>>cur.quantity;

cout<<"\nEnter Item Cost:"; cin>>cur.cost;

o1.push\_back(cur);

}

void print(Item &cur){

|  |  |
| --- | --- |
| cout<<"\n"; |  |
| cout<<"\nItem Name:"<<cur.name; | |

cout<<"\nItem Quantity:"<<cur.quantity; cout<<"\nItem Cost:"<<cur.cost;

cout<<"\nItem Code:"<<cur.code;

}

void display(){

for\_each(o1.begin(),o1.end(),print);

}

void search(){

vector<Item>::iterator p; Item cur;

cout<<"\nEnter Item Code to search:"; cin>>cur.code;

p=find(o1.begin(),o1.end(),cur); if(p==o1.end()){

cout<<"\nNot found.";

}

else{

cout<<"\nFound.";

}

}

void dlt(){

vector<Item>::iterator p;  
Item cur;

cout<<"\nEnter Item Code to delete:"; cin>>cur.code;

p=find(o1.begin(),o1.end(),cur); if(p==o1.end()){

cout<<"\nNot found.";

}

else{

o1.erase(p);

cout<<"\nDeleted.";

}

}

bool compare(const Item &i1, const Item &i2){ return i1.cost < i2.cost;

}

int main(){

int ch;

|  |  |  |  |
| --- | --- | --- | --- |
| do{ |  | | |
| cout<<"\n1.Insert"; | | |  |
| cout<<"\n2.Display"; | | | |
| cout<<"\n3.Search"; | | | |
| cout<<"\n4.Sort"; | |  | |

cout<<"\n5.Delete";

cout<<"\n6.Exit";

cout<<"\nEnter your choice:";

|  |  |  |  |
| --- | --- | --- | --- |
| cin>>ch; | |  | |
|  |  | | |
| switch(ch){ | | |  |
| case 1: | | | |

insert();

break;

case 2:

display();

break;

case 3:

search();

break;

case 4:

sort(o1.begin(),o1.end(),compare);

|  |  |  |
| --- | --- | --- |
| cout<<"\n Sorted on Cost!"; | | |
| display(); | |  |
| break; |  | |

case 5:

dlt();

break;

case 6:

exit(0);

}

}while(ch <= 6);

return 0;

}

* + 1. Write a program using STL list for Sorting and Result Calculation with user-defined records: Election Record (voter name, voter id, voter age, vote)

Ans : #include<bits/stdc++.h>

using namespace std;

class Voting{

public:

char voter\_name[10];

int voter\_id;

int voter\_age;

char votes;

};

int arr1[4]={0};

char arr2[4]={'A','B','C','D'};

vector<Voting> vot1; list<char> final\_votes;

void insert\_vote(){

Voting vot;

cout<<"\nEnter Voter's Name: ";

cin>>vot.voter\_name;

cout<<"\nEnter Voter's ID: ";

cin>>vot.voter\_id;

cout<<"\nEnter Voter's Age: ";

cin>>vot.voter\_age;

cout<<"\nEnter who you want to vote for out of(A, B, C, D) : ";

cin>>vot.votes;

vot1.push\_back(vot);

final\_votes.push\_back(vot.votes);

}

void print(Voting &vot){

cout<<endl;

cout<<"\nVoter's Name: "<<vot.voter\_name;

cout<<"\nEnter Voter's ID: "<<vot.voter\_id;

cout<<"\nEnter Voter's Age: "<<vot.voter\_age;

cout<<"\n"<<vot.voter\_name<<" voted for Candidate "<<vot.votes;

}

void display(){

for\_each(vot1.begin(), vot1.end(), print);

}

void calculate\_votes(){

for(auto it=final\_votes.begin();it!= final\_votes.end();it++){

arr1[int(\*it)-65]+=1;

}

}

void swap1(int \*xp, int \*yp){

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void swap2(char \*xp, char \*yp)

{

char temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void sort\_display(){

for (int i = 0; i < 4; i++){

|  |  |  |  |
| --- | --- | --- | --- |
| for (int j = 0; j < 4; j++){ | | |  |
| if (arr1[j] > arr1[j+1]){ | | |
| swap1(&arr1[j], &arr1[j+1]); | | | |
| swap2(&arr2[j], &arr2[j+1]); | | | |
| } | |  | |
| } |  | | |

}

for(int i = 3; i >= 0;i--){

cout<<"\nCandidate "<<arr2[i]<<" got "<<arr1[i]<<" votes";

}

cout<<endl;

}

int main(){

char ch='Y';

int choice;

cout<<"\t\nMENU\n"

|  |  |  |
| --- | --- | --- |
| <<"1. Vote\n" |  | |
| <<"2. Display Voters and their Votes\n" | |  |
| <<"3. Calculate the number of votes(necessary)\n" | | |

<<"4. Sort the Poll and Display no. of votes for candiates.\n"

<<"5. Exit\n";

do{

cout<<"\nChoose a number between(1-5): ";

cin>>choice;

switch(choice){

case 1:

insert\_vote();

break;

case 2:

display();

break;

case 3:

calculate\_votes();

break;

case 4:

sort\_display();

break;

case 5:

exit(0);

}

cout<<"\nDo you want to continue?: ";

cin>>ch;

}while(ch=='Y' || ch=='y');

return 0;

}

* + 1. Write a program using STL for Sorting and Hotel Bill Calculation with user-defined records: Hotel Bill Calculation record (items [], customer name, total amount, table no).

Ans :

#include <iostream> #include <algorithm> #include <vector> using namespace std; class BillRecord{

|  |  |
| --- | --- |
| public: |  |
| vector <int> item; | |

string cust\_name;

|  |  |
| --- | --- |
| int total\_amt; | |
| int table\_no; | |
|  |  |

bool operator==(const BillRecord& i1){

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| if(table\_no==i1.table\_no) | | | | |
| return true; | | | |  |
| return false; | | |  | |
| } | |  | | |
|  |  | | | |

bool operator<(const BillRecord& i1){

|  |  |  |  |
| --- | --- | --- | --- |
| if(table\_no<i1.table\_no) | | | |
| return true; | | |  |
| return false; | |  | |
| } |  | | |

};

vector<BillRecord> cust;

int price[5] = {0, 100, 200, 200, 400};

void insert(){

BillRecord cur;

cur.total\_amt = 0;

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

|  |  |
| --- | --- |
| cout<<"1.Pasta - 100 Rs\n"; |  |
| cout<<"2.Punjabi meal - 200Rs\n"; | |

cout<<"3.South Indian meal - 200Rs\n";

|  |  |
| --- | --- |
| cout<<"4.Fried Rice(veg) - 400Rs\n"; |  |
| cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"; | |

cout<<"Enter number of items to order :"; int n; cin>>n;

cout<<"Enter item number and quantitiy of all items : ";

|  |  |  |
| --- | --- | --- |
| for(int i=0; i<n; i++){ |  | |
| int x, y; cin>>x>>y; | |  |
| cur.item.push\_back(x); | | |

cur.total\_amt += price[x]\*y;

}

cout<<"\nEnter name: "; cin>>cur.cust\_name;

cout<<"\nEnter Table number:"; cin>>cur.table\_no;

cust.push\_back(cur);

}

void print(BillRecord &cur){ cout<<"\n";

cout<<"\nCustomer Name: "<<cur.cust\_name; cout<<"\nTotal Orders: "<<cur.item.size();

cout<<"\nTotal amount to pay:"<<cur.total\_amt; cout<<"\nTable number:"<<cur.table\_no;

}

void display(){

for\_each(cust.begin(),cust.end(),print);

}

void search(){

vector<BillRecord>::iterator p; BillRecord cur;

cout<<"\nEnter table number:"; cin>>cur.table\_no;

p = find(cust.begin(),cust.end(),cur); if(p == cust.end()){

cout<<"\nTable is empty.";

}else{

print(\*p);

}

}

bool compare(const BillRecord &i1, const BillRecord &i2){ return i1.table\_no < i2.table\_no;

}

int main(){

int ch;

do{

cout<<"\n\*\*\*\*\* Select a option \*\*\*\*\*\*\*\*\*\*\*";

cout<<"\n1.Place new order";

cout<<"\n2.Display All order";

cout<<"\n3.Display Bill of given table number :";

cout<<"\n4.Sort";

cout<<"\n5.Exit";

cout<<"\n\*\*\*\*\*888888\*\*\*\*\*\*\*\*\*\*\*";

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cout<<"\nEnter your choice:"; | | | | |
| cin>>ch; | |  | | |
|  |  | | | |
| switch(ch){ | | |  | |
| case 1: | | | |  |

insert();

break;

case 2:

display();

break;

case 3:

search();

break;

case 4:

sort(cust.begin(),cust.end(),compare);

|  |  |  |  |
| --- | --- | --- | --- |
| cout<<"\n Sorted on table number!"; | | | |
| display(); | | |  |
| break; | |  | |
| case 5: |  | | |

exit(0);

}

}while(ch <= 6);

return 0;

}

1. **References:**
2. **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**
2. **Tony Gaddis- “STARTING OUT WITH C++ From Control Structures through Objects”, Pearson Education, ISBN 13: 978-0-13-376939-5**
3. **| P a g e A r m y I n s t i t u t e o f T e c h n o l o g y , P u n e**