stack, queve, priority queve. * rectors are dynamic arrays which have the ability to resize itself dynamically when an element is inserted or deleted. * The storage of vector is handled atomatically by the container class. the general form of vector declaration is vector cobject-type>v1; Ext yector kint sa; a should illibed men of vector < float > f; Vector < int > $a = \{1, 2, 3, 9, 5\}$; * some of the member functions of vectors are: at(): Returns the element at the specified index 2 back(): Returns the reference to the last dement.

3 begin (): Returns an iterator pointing to the first element of the vector.

4 maximum number of 4. capacity(): Returns the maximum number of elements that are allowed. Line some prize

5. clear(): Deletes all the elements from the vector and assign an empty vector. 6. empty(): Returns a boolean value, true if the vector is empty and false if the vector is not empty. 7. end(): Returns an iterator pointing to the end of the vector. 8 erasect: Délètes a single dement or a range of elements: losti asies of the 9. Front (): Returns the reference to the first element. balbood at votor to spondled . Inemale vector at a particular position. 11. pop-back(): Removes the last element from the vector. 12 push-back(): Inserts a new element at the end of the vector. 13 resize(): Resizes the vector to the new length which can be less than or greater than the current length. 14. size(): Returns the number of elements in the vector somewhat and and and and and C++ program to implement vector and its functions: words and to transport the # include kiostream->com and a motor indiana # include < vector> bossollo sono fort strong using namespace std;

int mair int ii vecto vecto cout. for (VI. F cout for (cout cout for (cout ite COU 11. 11.

CO

CO

co

co

0

```
int main()
the
                                 F. Pitro is margin
          int vector zint > vi;
vector zint >:: iterator itr:
vector
          coutex" h vector is empty: "ex vi empty();
for (i=0; i < 5; i++)
the
          vi. push_back(i+i);
10335V
          cout ex "In vector elements are "; homes de
, a ...
          for(itr = v1. begin (); itr! = v1. end (); itr++) = +200
Philliese
          cout << *itr << "; " it washed to formats
first
          cout ex 'In vector elements are:";
3/17
          for(i=0;i<5;i+t)
motor
          cout < vi[i] << ";
          itr = v1. begin ();
from
          cout << "In first element: ' << * itr;
           v1. push-back(6);
           vi. pop-back();
           cout << "In first element: " << v1. front();
          cout <a" in last element: " << v1. back();
new
           itr = V1. end()-1;
eater
           cout << "In last element: " << * itr;
          couter in Element at index 4: " xx v1. at(4);
ents .
          cout ex" In capacity: " << VI. capacity();
150d 15
           cout ( "In size: ( < VI. size())
its
          Cout << 'In vector is empty: "<< 11 cmpty();
           return of the brook of the chipsel we still the
                                      · ( ) ripod · Iv : + ii
```

```
Nector is empty: 1
   Vector elements are: 123 45
Vector elements are: 123 45
   first element: 1
  tirst element: 1 (0 1-1) dood day to
   last element: 5 mm showeds whom of as they
  last element: 57 has 1v = 1 131; ( ) alpod 1v = 141) has
  Element at index 4:5
 capacity: 8 11 see stranged works of as him
 size: ±0.5

Vector is empty: 0

Chairman in the standard of th
 program: the set towned desit of se how
# include ciostream > (a) doed day in
# include < vectors (1) about 909 11
using namespace stali transle tent of as loss
int main (1) sood sures " thereals see at the book
                                vector (int's vi; " in the mans tent of " so too
      vector kint >:: iterator itr;
for (i = 0; i < 5; i++)
v1. push-back (i+1);
        cout x2 'In vector elements are 14;
       for (itr = v1. begin (); itr! = v1. end (); itr++)
       cout ex *itrec" ";
       itr = VI- begin();
      cout << " In first element: " << * itr;
```

VI. Push-Mr. Popcout ex cout ec ity = V1. coutee cost ex cost ex contiex cout 22 Villaser VI. inse vi. era VI-Jera cost K for (itr cout vi. cre cout <

for (it

Output:

first e

last e

last.

```
VI. push-back (6);
 VI. Pop-back ();
 v1. pop-back();
cout << "In first element: "<< v1. front ();
 cout < " In last element: " << 11. back();
 ity = v1. end()-1; end()-1; end()-1;
 coutec" In last element: "xx *itr;
 cout ex 'In Element at index 4: " < < v1. at (4)).
 cout ex" In capacity: " << 11. capacity();
 cout ex "in size!" xx vi size();
 cout ex "In vector is empty: " << vi empty();
 villasert (VI begin(1,0);
 v1. insert (v1. end(1,6);
 villerase (vil begin (1))
 VI-lerose (VI-begin()+1, VI-end()-1);
 cout KK "In vector elements are: ";
 for(itr=v1.begin(); itrl=v1.end(); itr++)
 cout < * Hitrock" (") a chop II : (1 hood houg )
VI. clear() dell on the
 cout « " In vector elements are:
 for (itr = v1. begin(); itr! = v1. end(); itr++)
 cout ex *itres she deletes the fast vite for
 return o's pape tout o's deletes the tiest clement
or not
Vector elements dren: 12/3 4 5 1 :1) 35/2 3
first element: 1 del ant oi tossarq
tirst elemential and that the in cosis wor
last element: 5
last element: 5
```

1 10

300%

1000

toos

) 101

7.77

3000

toos

110

1000

* 37

1000

1120

-12

1000

000

1 37

000

tuos

000

lix00

y si

9. back (): Element at index 4:5 (U) 1000-909 1V Size: 5 (1) thorn IV ss "thomals tent of ss June Capacity: 8 10. swap () Vector is lempty: 0 = "transle deal of set on 11. reverse Vector elements are : 1 6

Vector elements are:

Lists: 12. sort (an incre 13. merge nout ex " for coposity: " < VI. coposity(1) to 14. begin * Lists are sequence containers that allow first el non-contiguous memory locations for elements. 15. end * list containers are implemented as doubly. linked list so it is possible to add and . end o remove elements from both ends of the list. 16. cleo * lists are efficient to insert new elements. list a and to sort and mergers lists is our of so to M. era * some of the member functions of historie; rarge 1. push-back(): It adds a new element at the Program 2. push-front(): It adds a new element to the front. end of the list # inclu # incl using 3. pop-back(): It deletes the last element int m 4. pop-front (): It deletes the first element 1 list 5. empty(): It checks whether the list is empty 6. size(): It finds the number of elements present in the list. 7 max-size(): It finds the maximum size of the list.

8. front(): It returns the first element of the list 7.12 q. back(): It returns the last element of the list 3600 10. swap(): It swaps two lists oil and there is tung 11. reverse(): It reverses the elements of the list * + to 12. sort(): It sorts the elements of the list in 3400 an increasing order. 3401 is merge(): It merges the two sorted lists. llow. 14. begin(): Returns an iterator pointing to the intso. first element of the list. bly -15 end (): Returns an iterator pointing to the d 17. end of the list. (Classification) · slist. 16 clear(): Deletes all the elements from the ents. list and assign an empty list. JU53 14 erase(): Délètes à single élément or à ive !! range of elements. Program -10 - 11 #include kiastream > , the # include < list > " !! using namespace std; int main () + (() Lho 1) = 1 + (() orgod 1) = +1) + or 2000 10331 list <int>:: iterator itr; l list kint > L1, L2; empty 11. push-back (5); 11. push-back(10); who showed of so do 11. push-back(15); · ! · di ; () mipod · 11 = · di) voi 20 11 push-back(20); 12. push-back (70);

```
101 (2 push-back(80); 11 =11 2000) 11 :() 1
     12. push-back (90); 101 201 2010 11 11 11
                                                out put:
     (2. push-back(100);

Cout ec "In list elements are:;

for (itr = LI. begin 1); itr! = LI. end(); itr++)
                                                I vist e
                                                   Eleme
     cout << * itr << "; "; police out of 2000 prizeson
                                                · Cist 1
                                                IL List
                                                   eleme
     11. push-front (25);
                                                   Elemer
     ct. pop-front();
    11. push-back (30);
                                                  Maps
                                                 * Maps
    cout co 'In Elements in reverse order: ";
                                                  elemen
    11 reverse (); 31 phone no apiero 600 1
                                                 * Each
    for (itr = 11 begin (); itr! = 11 end(); t + itr)
                                                  mapped
      cout <c * itr «";
                       ege er elements
                                                 have
                                                 * The
                                                  can 1
 I (cout ex 'In list! elements after swapping:
                                                 * Hen
   11. swap (12); (12)
                                                  by . th
   for (itr = 11. begin(); itr ! = 11. end(); ++itr);
                                                  direct
                                                  * som
      cout << * itr << "; it who still the total
                                                  1. at ()
                                                  2. beg
                             (1. pash-back (5);
I cost << 'In Elements after merging; 1; 11.

11. merge (LL);
                                                  elemen
                                                 3. lend
                              ((ct) spect deep 1)
                                                  water
  for (itr = Lt. begin (); itr! = (10 end (); ++itr)
                                                 4. size
                              (OF) DOOD DUT !!
    cout ex *itr ex";
```

return Oin marine and set sounded : (1) to sall respect to the set of the output: prist elements are: 5 10 15 20 glements in reverse order: 20 15 10 5 just 1 elements after swapping: 70 80 90 100 elements in reverse order: 20 15 10 5 Elements after merging : 20 15 10 5: 70 80 90 100 Maps: * Maps are associative containers that store elements as sequence of (key, value) pairs * Each element has a key value and a pris mapped value. No two mapped values can have some key values. * The datatype of key value and mapped value can be different it is motorate it posts inis gom * Hements in map are always in sorted order by their corresponding key and can be accessed directly, by their key, this in I disease the best * some basic functions associated with 1 at (): returns value at the key. 2 begin(): returns on iterator to the first element in the maps is end(): returns on iterator to the end of the maps 6,0002 2 - +11 20 " 3/" >> 120/2 - +11 >> 100 4. size(): Returns the noiof elements in the map.

5. max-size(): Returns the maximum number of elements that the map can hold. 6 empty(): Returns whether the map is empty + insert (key, value): Adds a new element to the map. ? or at och ishis saistly in chamsing 8. clear() - Removes all the elements from the ma 9. erase (): Removes the element at the pastion pointed aby the iterators of server of strangly 10. swap (): Exchange the contents of the maps trogram: # include diastream > no quitaisosso sio equita # include & map > () do somepose do shomb RNO using namespace staged a and Immob doods int main() solor boggon out out salary boggon have some key values. map kinty string to student; and to say to be and map kint, string >:: Iterator, itr; hospillib and on Student insert (pair eint, string > (1201," bhavani")); student insert (pair <int, string > (1202, "kavya")); student insert (pair < int , string > (1205, "satya")); student insert (pair kint, string > (1205, "sunita")); student insert (pair cint , string > (1204, " renuta")) couter in Map Elements are: Initial company cout ex " In RNO IF NAME IN! for (itr = student . begin (); itr ! = student end(); itr cout << itr -> first << "It" << itr -> second exercly cout ex in Map size is: « student size ();

cout ec 10 c it student cout 22 1 ity -> secon itr = studer tr -- ; cost ee"In itr->secon return of

output;

Map Elem

1201

1202

1203

1204

1205

Map size student student

student

20000

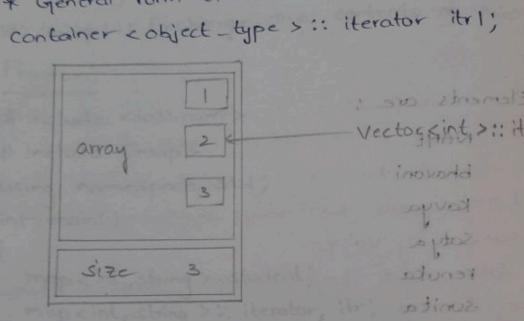
```
jout ec "In student with mo 1208: " < student at (1201)
of
      it = student . begin ();
      cout 22 'In student with rno!" << itr -> first << is "<
      itr -> second;
Tim
      itr = student · end ();

itr = -;

itr = -;
orthe .
      cost ex" In student with mo! " << itr -> first << is "<< itr -> second;
Just
costi
       return of iteration or contesson to mod contesson
      container & object . type > :: Heroton , ital; portain &
ops.
Me
      output:
       Map Elements are:
MX
                 NAME
1013
       RNO
                 bhavani
3 %
       1201
                 Kavya
       1202
                 satya
        1203
NOW!
                renuka
       1204
4下来
                                             - : 2miliosp18
                sunita
        1205
        Map size is: 3
                                         + STE provide
)); *
       student with rno 1203: satya
       student with mo: 1201 is bravani
); bd
        student with rno! 1205 is sunita
); v:
                                maintano dono decention y
9));
        swar skivary androppo bashnots not motorgo
2))/
                          ex amilian explains in Laborates
          to so born of mos mollows with more
```

Iterators !- se to soll one dies freshed and is () ripod . trobede * Iterators are pointer-like entities, that are used to access individual elements in a container.

* Iterators are moved sequentially from one element to another element. This process is known as iterating through a container * General form of iterator for a containers



Elements are ! vectogsint > :: iteratory inovodd Karya

renuka

syto2

Algorithms: -

* STL provide different types of algorithms that can be implemented upon any of the container with they help of iterators in the * Atthough each container provides its own basic operations, the standard algorithms provide more extended or complex actions. * Algorithm functions can be used on any type of container. * They allow us to work with two different typo

of containers at the same time.

Non-modifyin do not chan in the co modifying o designed t a containe pernoval a removie (Mutating (modifying specifically sorting o modifying efficient sorted r are spec function accordin This all Numeric designed Non-wa count() min - el

max-e

Search

coop << + 164 << - 63 5 VI End(); H. () 603 1V = المخافدون الله ١١٠ (١٤٠) ١٥٠ min_element()

mox_element()

fill() - 2 militagle pripition : 2 militagle pripition-non designed to work on numerical adolact as banquish Numeric algorithms: Numeric algorithms are This allows for greater efficienty. This will according to a particular sorting criterian. function on a container which is already sorted. or bangisab rudting by prition loising so sorted range algorithms. Dorted range algorithms. efficient sating at elements in a container. noditying algorithms specifically designed for sating algorithms: sorting algorithms are specifically to modify the order of elements. haditying algorithms, buit they are designed Mutating algaithms: mutating algorithms are remove elements in a container. Removal algorithms. These are designed to designed to alter the volve of elements within you not change the value lorder at any clement in the container. Modifying algorithms are noditying algorithms: Non modifying algorithms!

710 715

solig

aft.

150

1885

6/506

ata

Le gra

15× -

4.

· 500

. Ci 2

-

```
sorting ralgorithms ;
   Removal algorithms:
                         sort () 11- sound
   remove ()
                         stable-sort()
   Mutating algorithms:
                         make-heap ()
                         push-heap ()
   reverse ()
                         pop-heap()
   rotate()
                         sort-heap()
  Numeric algorithms:
                          sorted Range algorithm
                          binary-search()
   accumulate ()
                          tower-bound()
   partial_sum ()
                         upper-bound ()
                        existedly to modify the
            algorithms - sorting algorithms, o
#include kiostream >
                   pleasing and ropto point
# include < vector > de donnels to pritios."
# include Kalgorithms
# include, & numeric > the fundinople spine
using namespace std; methops priving lains
int main (4) solo si doides romintoss so
to a portion serting criterion.
  vector kint >. VI jointles releases not existes elle
   vector xint > !! iterator sitrin - and trop to sime!
   VI. push-back (10);
   vi. push-back (40);
   VI. push-back (20);
   VI. push - back (30);
   cout ex" In vector, elements are "invola
  tor (itr = v1. begin(); itr ! = v1. end(); itr+t)
  cout << * itr<< ";
```

sort (VI.b 11 reverse cout ex " for (itr = v cout ex cout < < " coutex cout xx' cost KK cout «« cout ec if (bina cout else cout < return

output:vector
after s
10 20
Maximum
Minimum

The s

Elemen

```
sort (v1. begin (), v1. end());
   11 reverse (vi. begin(), vi. end());
   cout ex "In after sorting: In";
   for (itr = v1. begin (); itr ! = v1. end (); itr + +)
   cout ex *itr ec" ";
   coutex" In Maximum element of vector is: ";
   cost << * max_element (VI. begin(), VI. end());
   cout << "In Minimum element of vector is:";
   cout KK * min_element (VI. begin (), VI. end (1);
   cout << "In The sum of vector elements is: ";
   cout ec accomulate (vi. begin(), vi. end(), o);
   if (binary - search (VI. begin(), VI. end (), 10))
   couter "In Element found in the vector";
   else
   cout ex" In Element not found in the vector";
   return o;
output:
vector elements are: 10 40 20 30
after sorting:
10 20 30 40
Maximum element of vector is: 40
Hinimum element of vector is: 10
The sum of vector elements is: 100
Element found in the vector.
```

-

hims:

tem

2000

141

+011

New York

70/1