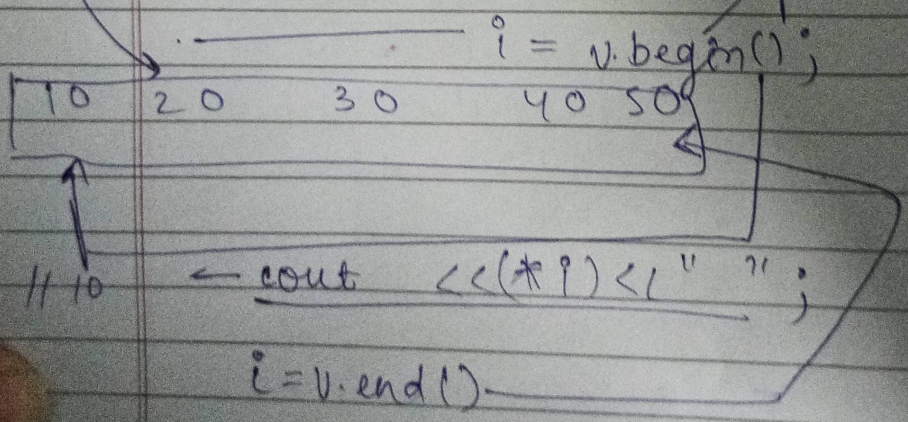
# STL C++

Containers (i) Simple → pair, vector, forward_list, list
(ii) Adapters → stack, queue, priority queue
(iii) Associative → set, map, unordered_set, unordered_map

## Iterators

* vector<int>::iterator i;
  OR
  auto

→ Iterator are exactly not treated as pointer is any object that, pointing to some element in a range of elements (such as an array or a container), has the ability to iterator through element of that range.



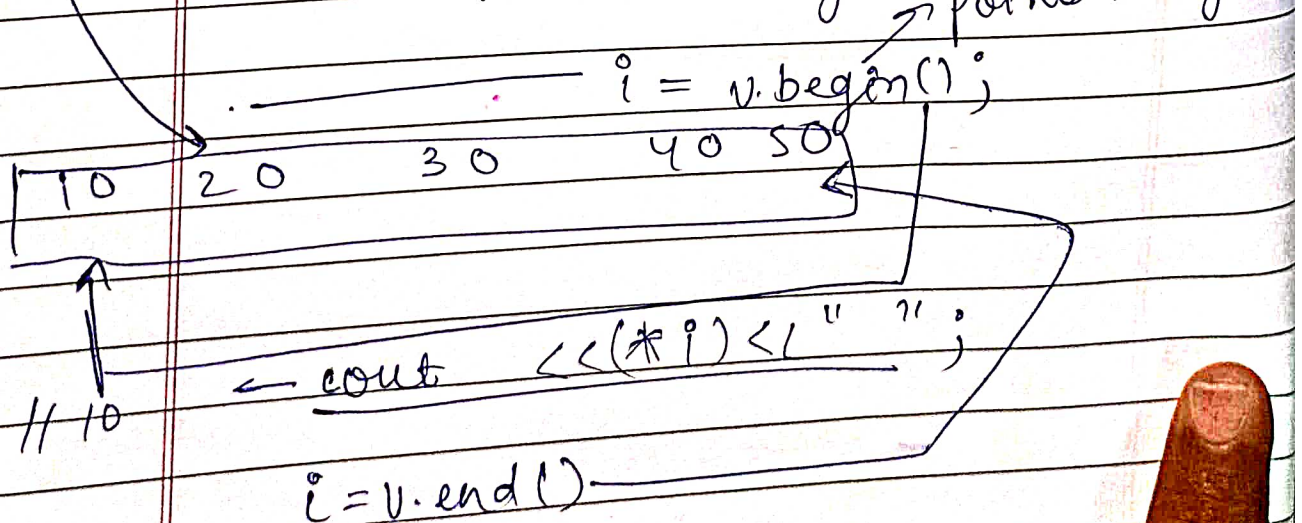
# STL C++

Containers (i) Simple → pair, vector, forward_list, list
(ii) Adapters → stack, queue, priority queue
(iii) Associative → set, map, unordered_set, unordered_map

## Iterators

* vector<int>::iterator i;
OR
auto

→ iterator are exactly not treated as pointer is any object that, pointing to some element in a range of elements (such as an array or a container), has the ability to iterator through element of that range.



cin.ignore() → after taking string input

RANKA
DATE / /
PAGE

fixed << setprecision(1) << (d+x) << "\n";

→ next function is used to get iterator to next no. a head & previous function gives previous value..

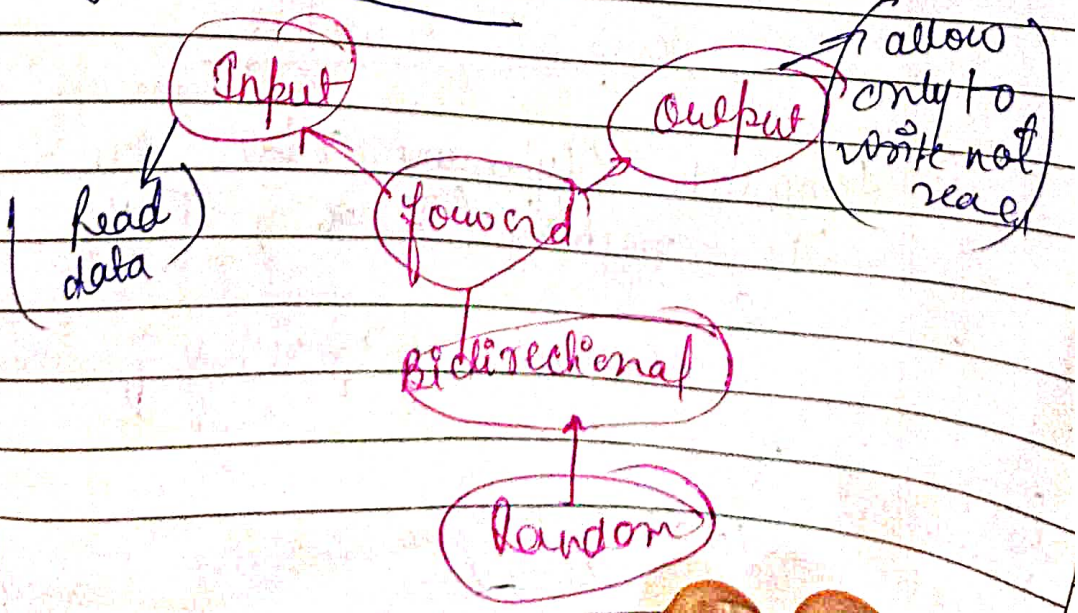
```
vector<int> v={10,20,30,40,50};
iterator i = v.begin();
i = next(i)
cout<<(*i)<<" "; → 20
i=next(i,2);
cout<<(*i)<<" "; → 40
i=prev(i)
→ 30
```

Advance function →

advance(i, 3) → moves 3 positions a head
cout<<(*i)<<" ";

(output → 40)

5 types of iterators →



Iterators

Simple containers
- → forward_list → forward
- → list → Bidirectional
- → vector → random

Associative containers
- → set
- → map
- → multimap
- → multiset

→ Bidirectional

- → unordered_set
- → unordered_map

→ forward

Adapters
- → queue
- → stack
- → priority queue

→ no iterators

## Templates in C++

→ like macros, processed by compiler.
→ Two templates → ① function template
② class template

## Template Example

```
template <typename T>
T myMax (T x, T y)
{
    return (x>y)? x: y;
}
```

T → type of data

for every datatype (datatype)

RANKA
DATE / /
PAGE

int main()
{ cout << myMax<int>(3,7) << endl;
cout << myMax<char>('c','g') << endl
}

different datatypes ke liye template

- Macros doesnot do type checking
- possibilities of error
- difficult to debug

Macros → #myMax2(x,y) ((x)>(y)? (x) : (y))

① Function Template
Write function ones let compiler generate the things according to data type.
eg → sort, linear search etc - - -

② Class Template (Stack, queue, deque etc.)

template <typename T>
struct pair {
T x, y;
pair(T i, T j) {x=i; y=j;}
T getFirst() { return x; }
T getSecond() { return y; }
};

(next page)

can also be written outside the struct.

```
int main()
{
    Pair <int> p1 (10,20)
    cout << p1.getfirst()<< " ";
    cout << p1.getSecond();
    return 0;
}
```

```
templat <type name T>
Pair<T> :: getfirst()
{
    return x;
}
```

outside class.

## Pairs in c++ STL

(two values stores, same or different type)

- pair<int, int> p1(10, 20);
- pair <int, string> p2 (10, "Geeks for Geek");
- cout << p1.first << " " << p1.second << endl
  << p2.first << " " << p2.second << endl

  → 10 20
  → 10 Geeks for Geeks

- By default empty pair has zero value.
  or
  p1 = {10, 20}
  (initialization)

function (ma
p1 = make_

comparison
pair<int,
cout <<
<<
<<
<<

only compares first value

# Sort one
(done

```
int main()
{
    Pair <int> p1 (10,20)
    cout << p1.getfirst()<< " ";
    cout << p1.getSecond();
    return 0;
}
```

```
template <type name T>
Pair<T> :: getfirst()
{
    return x;
}
```

outside class.

## Pairs in c++ stl

(two values stores, same or different type)

- `pair<int, int> p1(10, 20);`
  `pair <int, string> p2 (10, "Geeks for Geek");`
  `cout << p1.first << " " << p1.second << endl`
  `<< p2.first << " " << p2.second << endl`

  → 10 20
  → 10 Geeks for Geeks

- By default empty pair has zero value.
  or
  `p1 = {10, 20}`
  (initialization)

function (make_pair)

```
p1 = make_pair(10, 20)
```

RANKA
DATE / /
PAGE

Comparison Operators :

```
pair<int, int> p1 (1, 12), p2 (1, 12)
cout << (p1 == p2) << " "
     << (p1 != p2) << " "
     << (p1 > p2) << " "
     << (p1 < p2) << "end";
```

only compares first value

# Sort one Array Acc. to other. (done in function section).