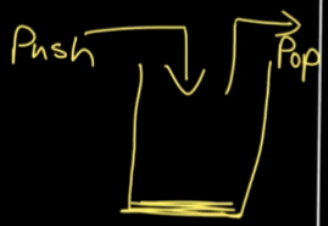
Implementing QUEUE using SINGLY LINKED LIST

Linked List vs Arrays

Advantages of Linked List over Arrays -

1) Dynamic size

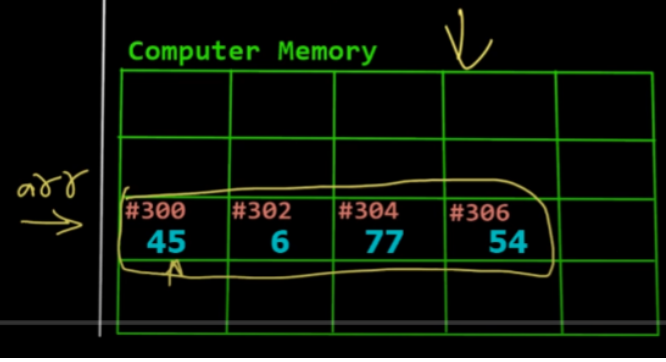


→ Queue using Array

```
int arr[4] = {45, 6, 77, 54}
```

First In First Out

Computer Memory



Implementing QUEUE using SINGLY LINKED LIST

Node Class -

```
class Node
{
  Data members -
int key // unique identifier
int value // actual value
Node *next // pointer to next node
  Member Functions (methods) -
   Node()
   {
     key = 0;
     data = 0;
     next = NULL;
   }
->Node(int k, int d)
   {
     key = k;
     data = d;
   }
};
```

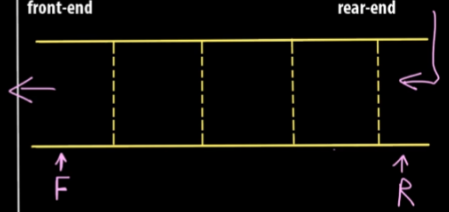
Queue Class -

```
class Queue
{
 Data Member -
-> Node *front    // (pointer in C++ )
-> Node *rear
 Member Functions(methods) -
-> isEmpty()      // check if queue is Empty
-> enqueue()      // add new node from rear end
-> dequeue()      // remove node from front end
-> count()        // get no of nodes in queue
-> checkIfNodeExist(Node n)
     // check if node with same key exist
-> void display()  // print all nodes in queue

}
```

Queue using Singly Linked List

Node (key, value , next)



Computer Memory

#100	#102	#104	#106
#200	#202	#204	#206
#300	#302	#304	#306

Implementing QUEUE using SINGLY LINKED LIST

Main Function -

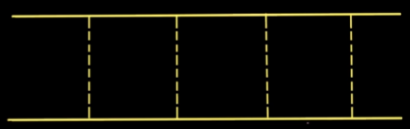
```
int main() {
  Queue q
  Node n1(1, 45),
    n2(3, 6), n3(7, 77),
    n4(2, 54)
  q.isEmpty()
  q.enqueue(n1)
  q.enqueue(n2)
  q.enqueue(n3)
  q.count()
  q.dequeue()
  q.dequeue()
  q.count()
  q.enqueue(n4)
  q.count()
  q.display()
}
```

```
isEmpty()
{
 if (front==NULL && rear==NULL)
    then
   return TRUE
 else
     return FALSE
}
```

Queue using Singly Linked List

Node (key, value , next)

front-end rear-end



F → NULL NULL ← R

Computer Memory

Implementing QUEUE using SINGLY LINKED LIST

Main Function -

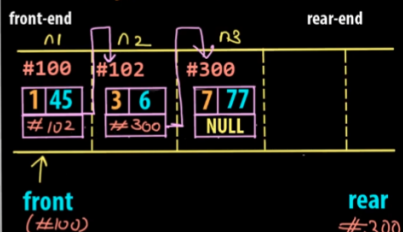
```
int main() {
  Queue q
  Node n1(1, 45),
  n2(3, 6), n3(7, 77),
  n4(2, 54)
  q.isEmpty()
  q.enqueue(n1)
  q.enqueue(n2)
  q.enqueue(n3)
  q.count()
  q.dequeue()
  q.dequeue()
  q.count()
  q.enqueue(n4)
  q.count()
  q.display()
}
```

```
enqueue(Node n)                n3(7,77)
{
1.if (isEmpty())
    then ->
   1.1 front = n
   1.2 rear = n
   1.3 print ("Node Enqueued")
2.else if(checkIfNodeExist())
    then ->
   2.1 print("Node already exist")
   2.2 print("Use different key")
3.else
   3.1 rear->next=n
   3.2 rear=n3
   3.3 print("Node Enqueued")
}
```

Queue using Singly Linked List

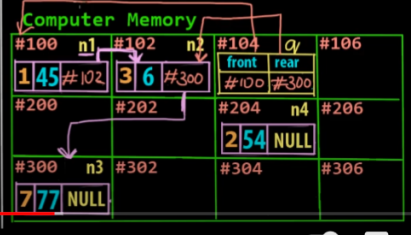
Node (key, value , next)

front-end rear-end



front (#100) rear #300

Computer Memory

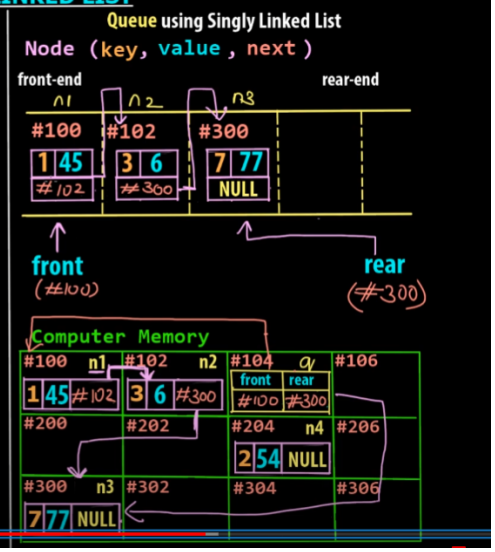


Implementing QUEUE using SINGLY LINKED LIST

Main Function -

```
int main() {
  Queue q
  Node n1(1, 45),
  n2(3, 6), n3(7, 77),
  n4(2, 54)
  q.isEmpty()
  q.enqueue(n1)
  q.enqueue(n2)
  q.enqueue(n3)
  q.count() // op -> 3
  q.dequeue()
  q.dequeue()
  q.count()
  q.enqueue(n4)
  q.count()
  q.display()
}
```

```
count()
{
 1.count=0
 2.Node *temp=front
 3.while->temp!=NULL
   {
   3.1count++
   3.2temp=temp->next
   }
 4.return count
}
```



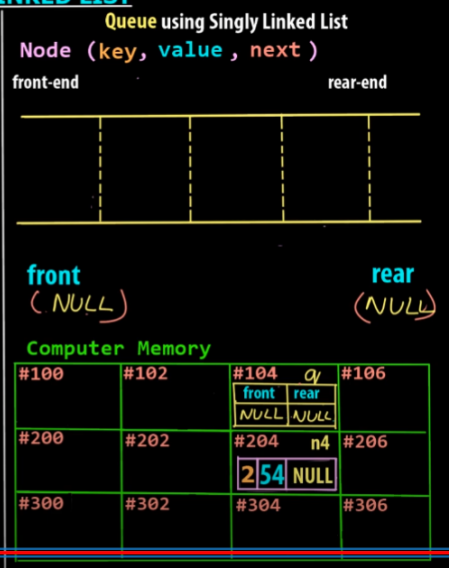
Implementing QUEUE using SINGLY LINKED LIST

Main Function -

```
int main() {
  Queue q
  Node n1(1, 45),
  n2(3, 6), n3(7, 77),
  n4(2, 54)
  q.isEmpty()
  q.enqueue(n1)
  q.enqueue(n2)
  q.enqueue(n3)
  q.count() // op -> 3
  q.dequeue() //
  q.dequeue() //
  q.count() // 1
  q.dequeue() // (7,77)
  q.enqueue(n4)
  q.count()
  q.display()
}
```

```
dequeue()
{
 1.Node *temp = NULL
 2.if(isEmpty()) then ->
   2.1 print("queue empty")
 3.else
   3.1 if(front==rear) then ->
       3.1.1 temp=front
       3.1.2 front=NULL
       3.1.3 rear=NULL
       3.1.4 return temp
     3.2 else
         3.2.1 temp=front
         3.2.2 front = front->next
         3.2.3 return temp
 }
```

temp = #300



Implementing QUEUE using SINGLY LINKED LIST

Main Function -

```
int main() {
  Queue q
  Node n1(1, 45),
  n2(3, 6), n3(7, 77),
  n4(2, 54)
  q.isEmpty()
  q.enqueue(n1)
  q.enqueue(n2)
  q.enqueue(n3)
  q.count()  // op -> 3
  q.dequeue() //
  q.dequeue() //
  q.count()  //  1
  q.dequeue() // (7, 77)
  q.enqueue(n4)
  q.count()  //  1
  q.display()
}
```

n5(2, 10)

```
checkIfNodeExist(Node n)
{
  1.Node *temp = front
  2.exist=false
  3.LOOP: while->temp!=NULL
   3.1 if temp->key == n->key
         then
       3.1.1 exist=true
       3.1.2 break
   3.2 temp=temp->next
    END LOOP
  4.return exist
}
```

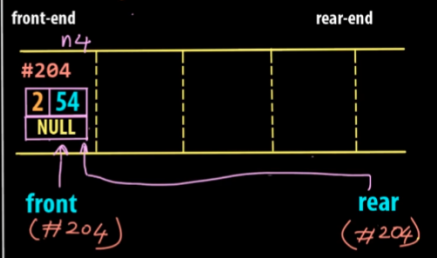
n5(2, 10)

2 == 2

temp = #204

Queue using Singly Linked List

Node (key, value, next)



Computer Memory

#100	#102	#104 q front rear #204 #204	#106
#200	#202	#204 n4 2 54 NULL	#206
#300	#302	#304	#306