Practical No:- 7

Title :- Write c++ program to maintain club members information using singly linked list.

Aim :- Department of Computer Engineering has student's club named 'Pinnacle Club'. Students of second, third & final year of department can be granted membership on request. Similarly one may cancel the membership of club. First node is reversed for president of club & last node is reserved for secretary of club. Write c++ program to maintain club member's information using singly linked list. Store student PRN & Name. Write functions to :-

(a) Add & delete the members as well as president or even secretary
(b) Compute total no. of members of club.
(c) Display members
(d) Two Linked lists exists for two divisions. Concatenate two lists.

Input :- Individual details

Output :- Maintain information of the Club member's.

Objectives :- To maintain club member's information by performing different operations like add, delete, reverse, concatenate on single linked list

Theory :-

linked list

- Not a sequential memory organization.
  - The noder need not be given adjacent memory locations
  - Instread, nodes can be placed anywhere in memory.

- Dynamic in nature
  - No. of nodes required need not be known at compile time
  - The list may grow or shrink during runtime

- New nodes can be added to this this list at any place.

- Existing nodes can be deleting from thist list.

Types of linked list

- Three major types of linked lists
  - Singly Linked list
  - Doubly linked list
  - Circular linked list

* Singly linked list

- A node from this list contains 2 factors
  - data can be int, char, float or an array also.

- Pointer
Holds address of next node

- Only only way traversal, from head to tail.
- No way to come back to previous node
- Address of 1st node should be remembered.
- Subsequent nodes are reached with node pointers

* Advantages
- Dynamic data structure
- No memory wastage
- Implementation
- Insertion & deletion operations.

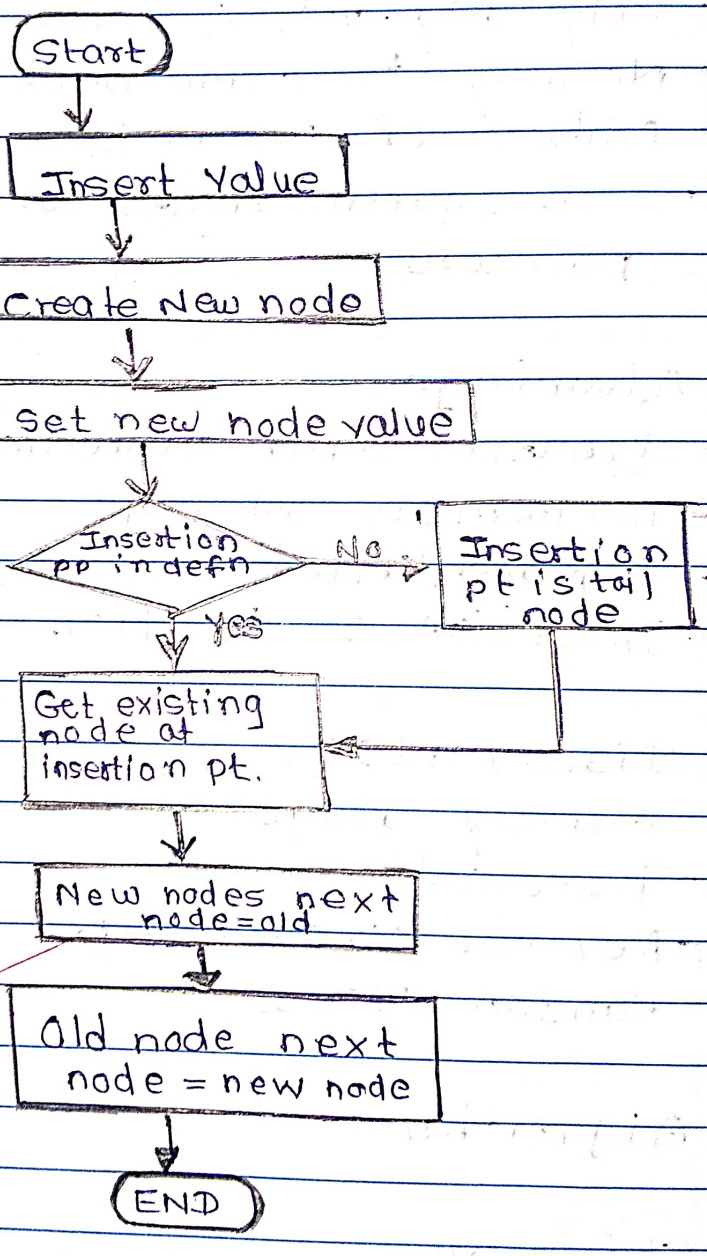
* Disadvantage
- memory usage
- Traversal
- Reverse Traversing
- Random access

Algorithm :-

St 1) Start
St 2) Define node structure by using class or structure
St 3) Define head & tail pointer assign to null inside the constructor.
St 4) Define method for create(), getnode(), insertnode(), deletenode(), display()

St 5) END

Flowchart :-



Conclusion :-

By this way, we can maintain club members information using singly linked list.

Practical No:-8

Title :-

Ticket booking system of cinemax theater has to be implemented using C++ program. There are 10 rows & 7 seats in each row. Doubly circular linked list has to be maintained to keep track of free seats at rows. Assume some random booking to start with. Use array to store pointers (Head pointer) to each row. On demand

a) The list of available seats is to be displayed.
b) The seats are to be booked
c) The booked can be cancelled.

Prerequisite :- Knowledge of doubly circular linked list. Representation of circular linked list. Knowledge of ticket booking

Objective :- To perform doubly circular linked list for cinemax ticket booking. To display available seats.
To book cancel & seats.

Input :- Row no. & seat no. to book seat.

Outcome :- Display available seats to book movie ticket.
Display status of Booked seat/cancel seat.

Theory :-

Circular Doubly linked list

- A node from this list contains 3 factors
  * data
  * pointers
  - Holds address of next node.
  - Holds address of previous node.
  - Its data type is 'struct node'
- Two way traversal, from head to tail & tail to head
- Feature : last node points to first node
- Address of first or last node should be remembered

Advantages

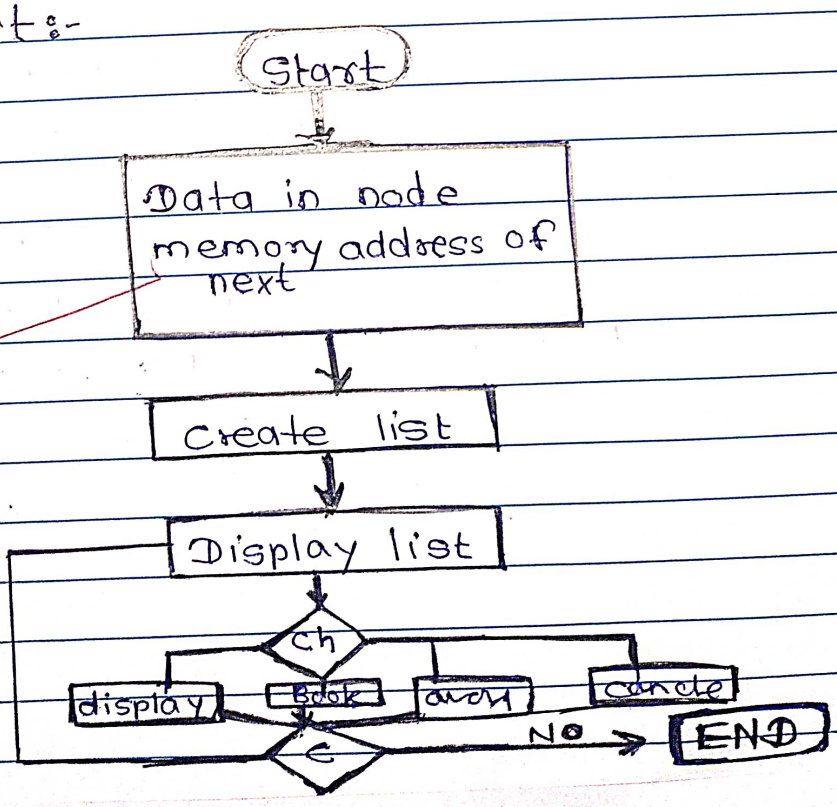
- list can be traversed from both directions i.e. from head to tail or from to head.
- Ease of data manipulation
- Jumping from head to tail or vice versa takes

Disadvantages
- Requires additional memory
- More complex than Singly linked list
- If not used properly, then the problem of infinite loop can occus.

Algorithm

St 1] Start
St 2] Create a linked list containing to elements in format 10X2
St 3] Declare & define class (Book)
St 4] Declare cancle class & define it
St 5] In main fun^n display updated matrix & create a loop for book & cauel class
St 6] STOP

Flow chart:-



Practical No. 9

Practical Title :- Write the ~~python~~ C++ program to perform string operations

Aim :-
Write a python program to compute following operations on string.

a] To display ;word with the logest length
b] To determine the frequency of occurance of particular character in the string
c] To check whether given string is palindrome or not
d] To display index of first appearance of substring
e] To count the occurance of each word in a given string

Pre-requisite :- Basics of string operation

Objectives :- To understand the use standard library functions for the string operations
To perform the string operations.

Input :- One or Two strings

Output :- Resulting string after performing string operation.

Theory :-

String :-

String is defined as an array of characters or a pointer to character.

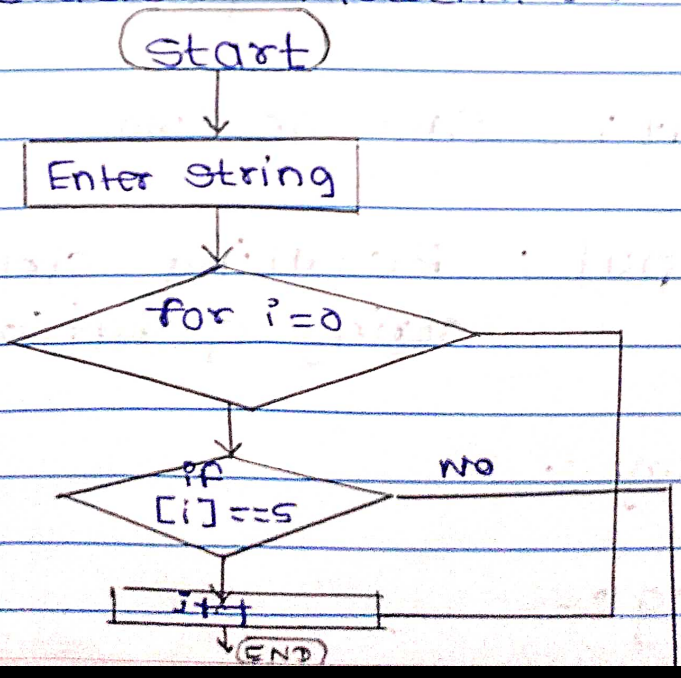
Null-terminated strings:-
String is terminated by a special character which is called as NULL terminator or null parameter (10) So when you define a string you should be sure to have sufficient space for the null terminator.
The null terminator has value 0.

Declaring string
As in string definition we have 2 ways to declare a string first way is we declare an array of characters as follows char s[] = "string";

or

char str[20];

String Operation Flowchart:-



Practical No. 10

Title :-
Write a c++ program using stack whether given expression is well parenthesized or not.

Objective :-
1) To check the given expression is parenthesized or not by using stack.

Problem statement :-
In any language program mostly syntax error occurs due to unbalancing delimiter such as ().
C++ program using stack to chech wethether given expression is well parenthesized or not.

Theory :-

Stack :-

Stack is a LIFO (last in first out) structure. It is an ordered list of the same type of elements.
A stack is a linear list where all insertion & deletions are permitted at only one end of list.
When elements are added to stack it grows at one end. Similarly when elements are deleted from shrinks at the same end.

Stack using array :-

1) Stack is a LIFO structure. Stack can be represented using array

2) A one D array can be used to hold elements of stack.

3) Another variable "top" is used to keep track of the index of the top most elements.

4) The following operations operations can be done on the stack by using array.

1) Initialization
2) Is empty condition
3) Is full condition
4) Push condition
5) Pop condition

Parenthesized :-
In the assignment created by us the meaning of parenthesized can be defined as in which statement in completed by using opening & closing brackets.

e.g.

1. (a+b)
2. {a+b}
3. [a+b]
4. {a[h+k (a+y)]}

Operation :-
We used a stack to complete the operation using parenthesize. When we give a opening bracket in the statement we push the bracket in stack & in the case of closing bracket we pop top most opening brackets & compare with the closing bracket.

1. If the brackets are equal to each other the case of valid statement gets printed in the test case.

2. If the brackets are unequal to each other the case of invalid statement gets printed in the case.

3. If the stack is not empty & a series of closing brackets has been exhausted then also statement is not well parenthesized.

4. If the stack is not empty & a series of closing brackets has been exhausted then also the statement is

JSPM
Page No.:

Practical No. 11

Title :-
Write a c++ program to simulating job queue by using array & linked list.

Objective:-
1. To study queue using array
2. To study queue using linked list.

Problem statement:-
Queues are frequently used in computer programming & a typical example to is creation of a job queue by an operating system.

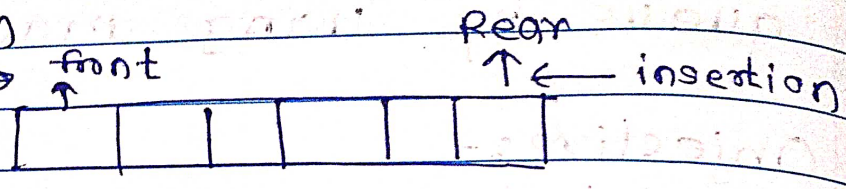
Theory & Concept:-

Queue

It is a special kind of list, where items are inserted at one end (the rear) & deleted from the other end (front). Queue is a FIFO (first in First Out) list. We come across the term queue in our day to day life. We see a queue at a railway reservation counter or a movie ticket counter. Before getting the service one has to wait in the queue

After receiving the service, one leaves the queue.

Service is provided at one end (the front) & people joint the other end (rear)



2. Queue using Array

An array representation of queue requires 3 entities.

a) An array to hold queue elements
b) A variable to hold the index of the ~~front~~ element.
c) A variable to hold the index of the rear element.

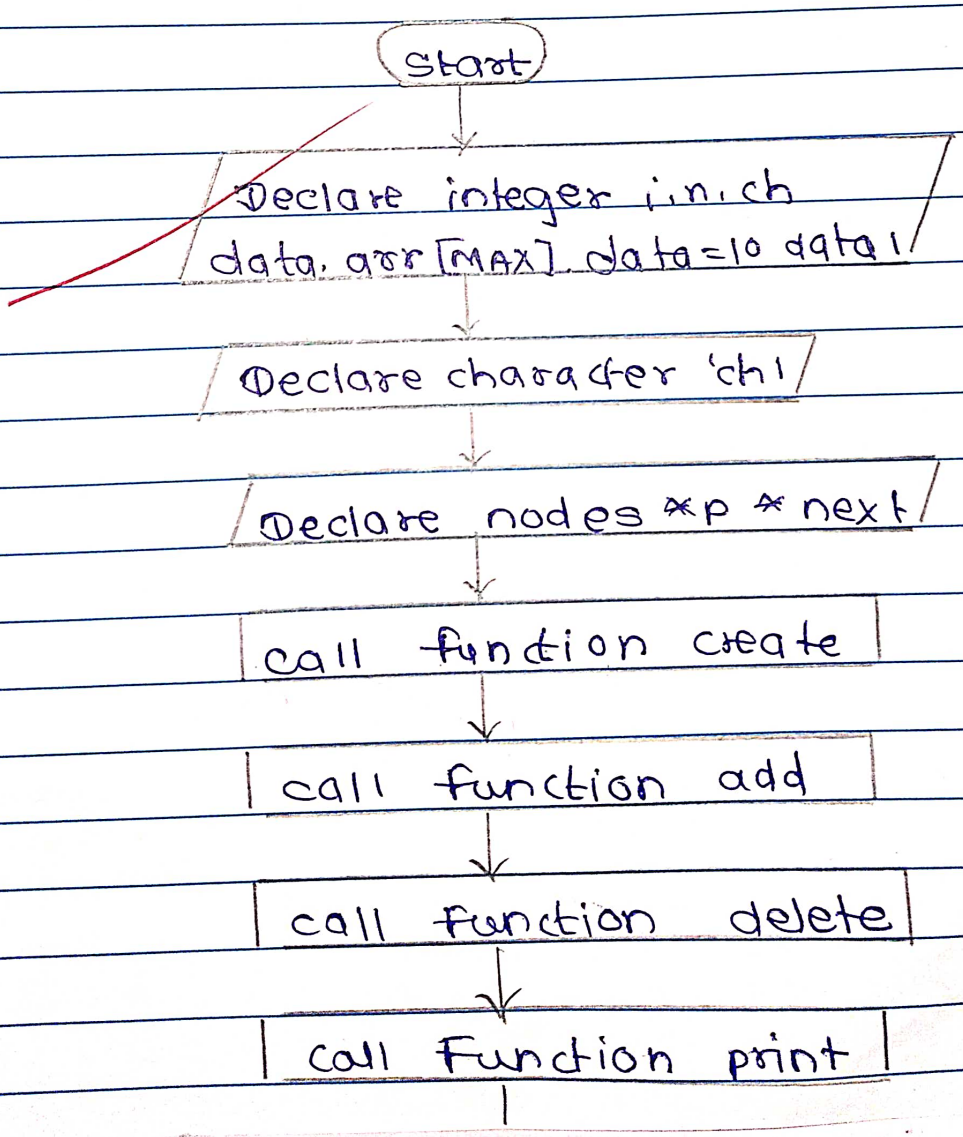
A Queue data type may be defined formally as follows:

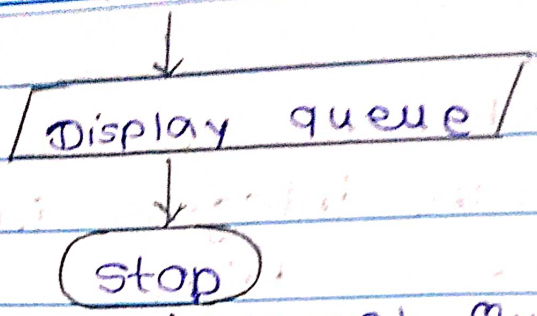
```
#define MAX 30.
type def struct queue
        {
        int data [MAX];
        int front, rear;
        } queue
```

Algorithms:-

St 1) Start
St 2) Declare integers i, n, ch, data, arr[MAX], data = 10, data1
St 3) Declare character ch
St 4) Declare nodes *p *next
St 5) Call function create
St 6) Call function add
St 7) Call function delete
St 8) Call function print
St 9) Display queue
St 10) Stop

Flowchart :-





Flowchart :- To implement Queue

Conclusion :-

Here with the help of queue using array & queue using linked list we successfully simulate a job queue

Practical No. 12

Statement :- A double-ended (deque) is a linear list in which additions & deletions may be made at either end.

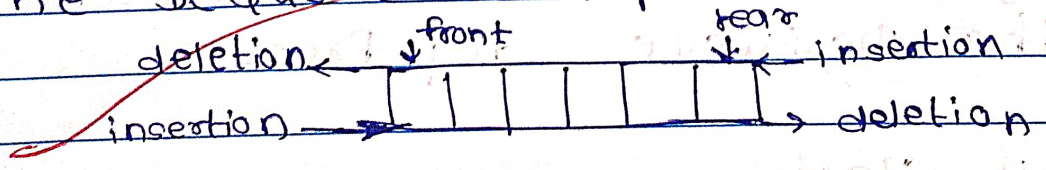
Objective :- Understand how to perform insertion & deletion in double ended queue.

Outcome :- Will be able to perform insertion & deletion in double ended queue.

Theory :- DeQueue is a data structure in which elements may be added to or deleted from the front or the rear. Like an ordinary queue, a doub-ended queue is a data structure it supports the following operations: end_front, enq_front, enq_back, deq_front, deq_back & empty.

Deque can be behave like a queue by using only enq_front & deq_front & behaves likes a stack by using only enq_front & deq_rear.

The dequeue is represented as follows.



Algorithm :-

Algorithm to add an element into DeQueue.

Assumptions:
pointer f, r & initial values are -1, -1 Q[] is an array max represent the size of a queue enq_front.

St 1) start
St 2) Check the queue is full or not
St 3) IF false update the pointer FAS
f = -1
St 4) Insert the element at pointer f as Q[] = element.
St 5) Stop enq_back.

Step 1): start
Step 2): check the queue is full or not as if (r == max - 1) if queue is full.
Step 3) if false update the pointer ras
r = r + 1
Step 4: insert the element at pointer r as Q[r] = element
Step 5: Stop

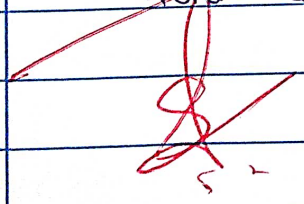
Algorithm to delete an element from the DeQueue deq_front

St 1) Start
St 2) Check the queue is empty or not as if (f == r) if yes queue is empty.

St 3) if false update pointer f as f = f+1 &
delete element = Q[f]
St 4) if (f==r) reset pointer f & r as
f = r = -1
St 5) Stop

deq-back
step 1) Start
step 2) check the queue is empty or not as
if (f==r) if yes queue empty
step 3) if false delete element at positio
r as element = Q[r]
step 4) Update pointer r as r = r-1
Step 5) if (f=r) reset pointer f & r as
f = r = -1.

Conclusion:-
In this way, we have simulated performed
addn & deln operations on Queue.



Practical No. 13

JSPM
Page No.:

Title :- Write a c++ program to implement queue.

Objective :- To add elements from rear end of Queue
To delete elements from front end of Queue.

Problem statement :-
Pizza parlor accepting maximum M orders. Orders are served in first come first served basis. Order once placed cannot be cancelled. Write a c++ program to simulate the system using circular queue using array.

Theory & Concepts :-
Double ended queue :- The word dequeue is a short form of double ended queue

It is representation of both stack & queue & can be used as stack & queue.

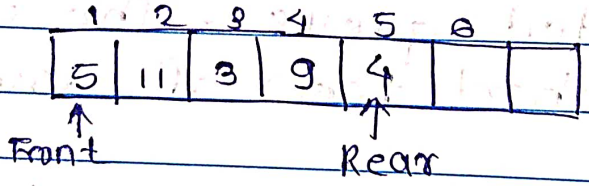
In a deque, insertion as well as deletion can be classified into 2 types :-

A] Input Restricated Deque :
The following operation are possible is an i/p restricated dequeue;
1) Insertion of an element at the rear end.

2) Deletion of an element from front end
3) Deletion of an element from rear end.

B) Output Restricted Dequeue:
The following operation are possible in an output restricted dequeue.
1) Deletion of an element from front end
2) Insertion of an element at rear end
3) Insertion of an element at front end

There are various methods to implement a deque.
a) Using a circular array
b) using a singly linked list
c) Using a singly circular linked list



Algorithm :-

St 1) Start
St 2) Declare integer array, integer front, rear
St 3) Declare integer data, ch
St 4) Accept integer array from user
st 5) Call function addf
St 6) Call function addr
St 7) Call function delf
St 8) Call function delr

st 9) Call function Print
st 10) Stop

Flowchart :-

start

Declare integer, Array, arr[],
integer front, rear

Declare integer data ch

Accept integer array arr[] from user.

Call function addr

Call function addf

Call function delf

Call function delr

Call function print

Stop

Conclusion :-
By using above code we successfuly implemented Circular Queue.