24-07-2024

Conditional compilization preprocessor directives :-

Always ends with # endif

```
# if        # ifdef
# else
#
```

```
-> main ( )
  { int a=5, b=6, c;
    # ifdef add
    c = add (a, b)
    printf ("%d", c);
    # endif
  }
```

Self referential struct

- Logical linear DS
singl linked list

if (front == -1)
front = 0
rare increase
insert element

deletion element
front increse

front = -1
queue empty
x = q [front]

for (i = front; i <= rear; i++)

APPLICATIONS OF STACK :-

-> Recursive ( ) calls

-> Infix to Postfix Conversion

-> Infix to Prefix Conversion

-> Postfix expression evaluation

-> Prefix expression evaluation

-> Trees & Graphs traversal

Self referential
struct

Logical linear DS
Some Linked List

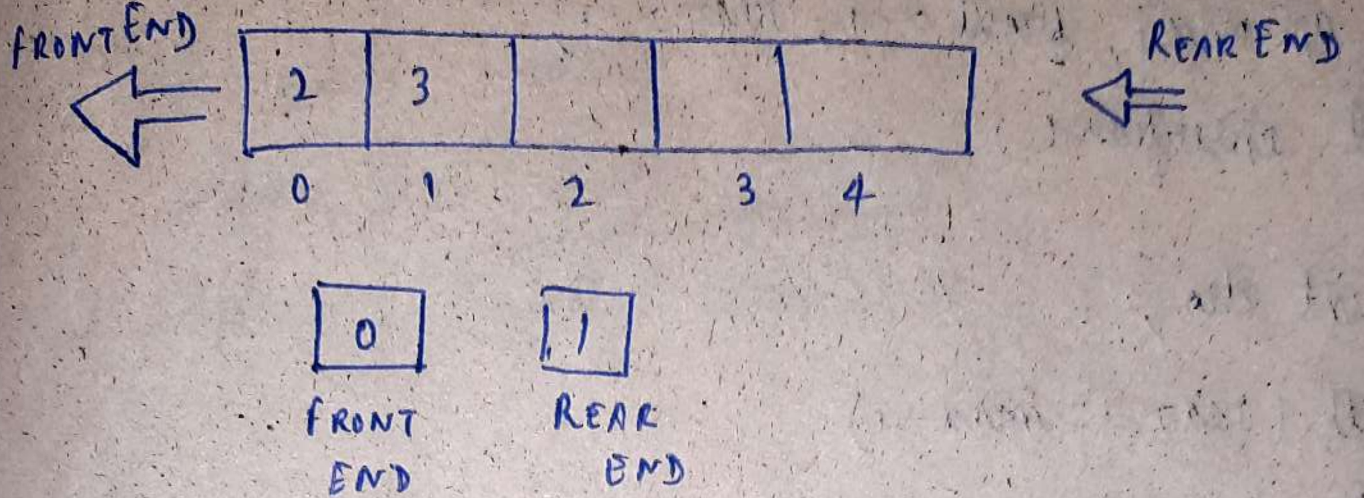
if (front == -1)
front = 0
rare increase
insert element

deletion element
front increase

front = -1
queue empty
x = q[front]

for (i = front; i <= rear;
i++)

QUEUE :-



-> Queue is a linear Data structure. In this elements are inserted from 1 end called Rear End & deleted from another end called Front End.

-> It operates on the principle of FIFO

-> FRONT is a pointer which always points 1st element in ~~the~~ a queue, whereas REAR is a pointer which always points last element in a queue.

OPERATIONS ON LINEAR QUEUE =>

1. INSERTION [enqueue]

2. DELETION [Dequeue]

```
-> # define max 5
int q[max], front = -1, rear = -1;
Void insertion ()
{
    int ele;
    if (rear == max-1)
    {
        printf ("full");
    }
    else
    {
        printf ("~~ele~~ enter ele ");
        scanf ("%d", &ele);
        rear++;
        q[rear] = ele;
        if (front == -1)
        front = 0;
    }
}
void deletion ()
{
    int x;
    if (front == -1)
    {
        printf ("empty ");
    }
    else
{
   k = q[ front ];
   if (front = = rear)
     front = rear = -1;  → else front ++;
     printf ("Delented element is %d", k);
   }
}
void display ()
{
   int i;
    for (i = front ; i <= rear ; i ++)
     printf (" %d", q[i]);
 }
main ( )
 {
   int ch;
   while (1)
   {
     printf (" enter 1 for insertion \n 2 for deletion \n
              3 for display of \n 4 for exit ");

     scanf ("%d", &ch);
     switch (ch)
      {
        case 1 : insertion (); break;
        case 2 : deletion (); break;
        case 3 : display (); break;
} }   }  case 4 : exit (0);
```