Unit - I

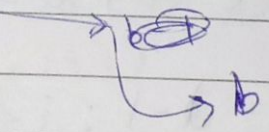
DATE: 29, 8, 23
PAGE:

■ Topic → Array Subtopic → 1-D Array

Q WAP to insert an element in the array

Sol →

```
#include <stdio.h>
#include <conio.h>

int main()
{int a[100]; int b; int i; int c; int d;
 clrscr();
printf("The length of/no.of element in array you
want :");
scanf("%d", &b); printf("Enter elements:");
for(i=0; i<b; i++)
{ scanf("%d", &a[i]); }

printf("Enter position at which you want to
insert element:");
scanf("%d", &c);
printf("Enter value:"); scanf("%d", &d)

for (i=b; i>c-1; i--)
{ a[i] = a[i-1];
}

a[c-1] = d;
return 0;
}
 for(i=0; i<=b; i++)
{printf("\n");
printf("%d", a[i]);
}

return 0;
}
```



Q. WAP to delete element & then shift.

Sol→

```
#include <stdio.h>
#include <conio.h>

int main()
{int a[100]; int b; int i; int d;
clrscr;
printf("Enter no of elements to enter :"); scanf("%d", &b);
printf("\n Enter Elements:");

for(i=0; i<b ;i++)
  { scanf("%d",&a[i];
    }

printf("\n Enter place which you want to vacant
        & delete value:");
scanf("%d", &d);

for(i=d-1; i<b-1; i++)
{ a[i]=a[i+1];
  }

printf("Displaying new array:");

for(i=0 ; i<b-1; i++)
 printf("%d", a[i]);

return 0;
}
```

Good Write

DATE: 31/8/23
PAGE:

Q WAP to merge two arrays.

Sol→

```
#include <stdio.h>
#include <conio.h>

int main()
{ int a[50]; int c[100];
int b[50];
int m,n; int i; int j; int p; k;
clrscr;
printf("Enter no. of elements you want in
first array: ");
scanf("%d", &m);
printf("Enter no of elements you want in
second array:");
scanf("%d", &n);
printf("Enter element in ascending order.\n");
printf("Enter elements in first array:");
for(i=0; i<=m-1; i++)
 scanf("%d", &a[i]);

printf("Enter elements in second array:");
for(i=0; i<=n-1; i++){scanf("%d", &b[i]);}

printf("\n Merging 2 arrays.\n");

for(j=0; j<m; j++)
 p= m + n;
```

Good Write

```
j=0; k=0;
for (i=0; i<p; i++)
{if (a[j] ≤ b[k])
   {c[i] = a[j]; j++;
   }
  else
   {c[i] = b[k]; k++;
   }
}
return 0;
}
```

Q. WAP to reverse an array

Sol→

```
#include <stdio.h>
#include <conio.h>

int main()
{int a[20]; int j; int m;
 clrscr();
 printf("Enter no. of elements in array: ");
 scanf("%d", &m);
 printf("Enter elements: ");
for (j=0; j<m; j++)
scanf("%d", &a[j]);

printf("\n Reversing array printing \n");
for (j=m-1; j>=0; i--)
printf("%d \t", a[j]);

 return 0;
}
```

Q. WAP to find sum of elements of array

Sol:-
```
#include <stdio.h>
#include <conio.h>

int main ()
{ int a[20] ; int i ; int m ; int b ;
clrscr();
printf ("Enter no of elements in array : ");
scanf ("%d", & m);

printf ("Enter elements : ");
for (i = 0 ; i < m ; i++)
  scanf ("%d", & a[i]);

printf (" Adding array elements. ");
 b = 0 ;
for (i = 0 ; i < m ; i++)
{b = b + a[i];
}
printf (" Addition is : ");
 printf ("%d", b);

return 0;
}
```

DATE: 31/8/23
PAGE:

■ Sub-topic → 2-D Array

2D Array

Q. WAP to give sum of elements of ~~2nd~~ matrix. (2D Array)

Sol→

```
# include <stdio.h>
# include <conio.h>

void main()
{ int a[50][50];
  int r, c, i, j; int sum=0;

  clrscr();

  printf("Input no. of ~~elements in~~ row in matrix:");
 scanf("%d", &r);

 printf("Input no of columns in matrix");
 scanf("%d", &c);
printf("Enter elements:");
 for (i = 0; i<r; i++)
{
 for (j= 0 ; j < c ;j++)
 {scanf("%d", &a[i][j]);
 }
}
printf("Array is :");
for(i=0 ; i < r ; i++)
{
 for( j=0 ; j< c ;j++)
 printf("%d", a[i][j]);
}
```

Good Write

```
printf ("\n Adding elements ");
sum=0.
for (i=0; i<r; i++)
{ for (j=0; j<c ; j++)
  sum += a[i][j];
}
printf ("\n Addition of elements is :");
printf ("%d", sum);

getch ();
}
```

Q. WAP to perform addition of matrix.

Sol→

```
# include <stdio.h>
# include <conio.h>
void main ()
{ int a[50][50] ; int b[50][50] ;
  int c[50][50];
  int r, c, i, j; clrscr ();

  printf ("Enter no. of row of matrix:");
  scanf ("%d", &r);
  printf ("Enter no of columns of matrix:");
  scanf ("%d", &c);

  printf ("Enter elements in 1st array:");
```

```
for (i=0; i<r; i++)
{ for(j=0; j<c ;j++)
  scanf("%d", &a[i][j];
}

printf("Enter elements in 2nd matrix:");
for (i=0; i<r; i++)
{
for (j=0; j<c; j++)
scanf("%d", &b[i][j]);
}

printf(" Adding both matrix");

for (i=0 ; i<r; i++)
{
for(j=0; j<c; j++)

c[i][j]= a[i][j] + b[i][j];
}

printf(" Displaying matrix :");
 printf("\n");

for(i= 0; i<r; i++)
{
for (j=0; j<c; j++)
{ printf("%d\t", c[i][j]);
}
printf("\n");
}
getch(); }
```

Good Write

DATE: __/__/____
PAGE: _____

Q. WAP to find sum of diagonal elements of matrix.

Sol→

```
#include <stdio.h>
#include <conio.h>

void main()
{ int a[50][50]
  int r,c,i,j;  int sum=0;

  clrscr();

  printf("Input no. of row & column in square
  matrix :");
  scanf("%d", &r);

  c = r;

  printf("Enter Elements :");
  for (i=0; i<r; i++)
  { for (j=0; j<c; j++)
    scanf("%d", &a[i][j]);
  }
printf("\n Matrix is :"); printf("\n");
for (i=0; i<r; i++)
{ for (j=0; j<c; j++)
  { printf("%d \t", a[i][j]); }
  printf("\n");
}
```

```
printf("\n Adding diagonal elements.");
printf("\n Processing");

for (i=0; i<r; i++)
{ for (j=0; j<c; j++
  { if (i==j)
    { sum = sum + a[i][j] ;
    }
  }
}
printf("\n")
printf ("Sum of diagonal elements is:");
printf ("%d", sum);

getch ();
}
```

Q. 2 WAP to perform sum of elements of upper triangular matrix.

Sol→

```
# include <stdio.h>
# include <conio.h>
void main()
{ int a[10][10];
  int r, c, i, j ; int sum = 0;

  clrscr();
  printf("Input no. of row & column in
    square matrix:");
  scanf("%d", &r);
  c = r;
```

```
printf("Input elements:");
for(i=0;i<r;i++)
 { for (j=0;j<c;j++)
 } scanf("%d", &a[i][j]);

printf("\n Adding Upper triangular matrix elements:");
);

for (i=0;i<r;i++)
{ for(j=0;j<c;j++)
 {if (j>=i)
  {sum = sum + a[i][j];
  }
 }
}
printf("\n sum of elements :");
 printf("%d", sum);

getch();
}
```

Lower triangular matrix element sum

* formula

```
if (j<=i)
{sum = sum + a[i][j];}
```

Q WAP to perform transpose of matrix.

Sol →
```
#include <stdio.h>
#include <conio.h>

void main()
{int a[10][10], b[10][10];
 int i, j, r, c;

 clrscr();

 printf("Input no of row in matrix:");
 scanf("%d", &r);

 printf("Input no. of column in matrix:");
 scanf("%d", &c);

 printf("Enter elements \n");

 for(i=0; i<r; i++)
 {for(j=0; j<c; j++)
  scanf("%d", &a[i][j]);
 }

 printf("\n Making transpose of input matrix\n");

 for(i=0; i<r; i++)
 {for(j=0; j<c; j++)
  b[j][i] = a[i][j];
 }
```

```
printf("\n Orginal Matrix is \n");

for(i=0; i<r; i++)
 {for(j=0; j<c; j++)
  {printf("%d \t", a[i][j]); }
  printf("%d
  printf("\n");
 }
```

```
printf("\n Transpose Matrix is \n");

for (i=0; i<c; i++)
{ for(j=0; j<r; j++)
  {printf("%d \t", b[i][j]); }
   printf("\n");
}

getch();
 }
```

Q: WAP to check if matrix is symmetric.

Sol→

```
# include <stdio.h>
# include <conio.h>
void main(){
int a[10][10], b[10][10]; int i, j, r, c;
 int flag = 0;
clrscr(); printf("Checking Symmetric or Square matrix.");
printf("Input no. of row in matrix :");
scanf("%d", &r);
printf("Input no. of column in matrix :");
 scanf("%d", &c);
```

```
printf("Enter elements");

for (i=0; i<r; i++)
{ for(j=0; j<c; j++)
   scanf("%d", &a[i][j]);
}

for( i=0; i<r; i++)
{ for( j=0; j<c ; j++)
   b[j][i] = a[i][j];
}

printf("\n Original matrix is \n");
for(i = 0 ; i<r; i++)
{ for(j=0 ; j<c ; j++)
  { printf("%d \t", a[i][j]); }
  printf("\n");
}

for (i = 0; i<r; i++)
{ for(j=0; j<c ; j++)
  { if (a[i][j] != b[i][j])
     flag = 1;
  }
}

if(flag == 1)
printf("\n Input matrix is not symmetric.");
else
printf("\n Input matrix is symmetric.");
getch();
}
```

Good Write

DATE: 4/9/23
PAGE:

Q. WAP to multiply two matrix.

Sol →
```
#include <stdio.h>
#include <conio.h>
void main() {
int a[10][10], b[10][10]; int m,n,o,p;
int i,j; int q; int c[10][10];

clrscr();

printf("Enter details of 1st matrix:");
printf("Row:"); scanf("%d", &m);
printf("Column:"); scanf("%d", &n);

printf("Enter details of 2nd matrix:");
printf("Row:"); scanf("%d", &o);
printf("Column:"); scanf("%d", &p);

if (n != o) {
   printf("Multiplication not possible."); }

else {
   printf("Enter elements of 1st matrix:");
   for(i=0; i<m; i++) {
    for(j=0; j<n; j++)
    scanf("%d", &a[i][j]);
   }
  printf("Enter elements of 2nd matrix:");
  for(i=0; i<o; i++){
   for(j=0; j<p; j++)
   scanf("%d", &b[i][j]);
}
```

Good Write

```
printf("Multiplication of both matrix.\n");

for(i=0; i<m; i++)
{for (j=0; j<p; j++
{ a=0;
  for(k=0; k<n; k++)
  { a+= a[i][k]*b[k][j];}
  c[i][j] = a;
}}

printf("\n Displaying multiplied matrix.\n");

for(i=0; i<m ; i++)
{ for(j=0; j<p; j++)
  {printf("%d\t", c[i][j]);}
  printf("\n");
}

printf(" New matrix is of order :");
printf("%d x %d", m, p);
} return 0;
  }
```

• Stacks

→ It is also linear data structure.

→ Based on LIFO.

Use
→ For reversing ~~a~~ name
→ Used in web browser.
→ Evaluating Expression

• Infix operator

a + b

Prefix

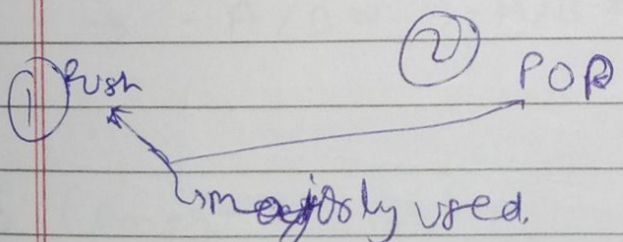
+ ab

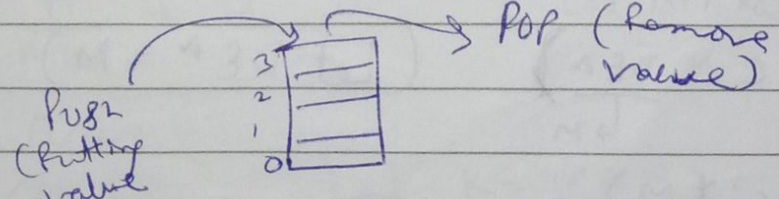
~~Suffix~~ Postfix

ab+

★ Q Reason behind requirement of prefix & postfix over infix.

Operations used in stacks





③ TOP (ds = -1 when stack is empty.)
(Good Write
Position) value till where stack filled,

④ ~~IS TRUE~~ ISFULL

■ Topic
Infix
Prefix
Postfix

⇒ Inspection & Hand Method

DATE: 6/9/23
PAGE:

→ Prefix & Postfix Question (came in University Paper)

Q. $A + [(B+C) + (D+E) * F] / G$
solve in Postfix.

Sol: $A + [(B+C) + \overset{\rightarrow Z}{\overbrace{DE+}} * F] / G$

$A + [(B+C) + Z * F] / G$

$A + [\overset{\rightarrow Y}{\overbrace{BC+}} + \overset{\rightarrow N}{\overbrace{ZF*}}] / G$

$A + [\overset{\rightarrow M}{\overbrace{YN+}}] / G$

$A + \underset{\rightarrow K}{\underbrace{MG/}}$

$AK+$

$AMG/+$
$AYN+G/+$
$ABC+ZF*+G/+$
$ABC+DE+F*+G/+$ Ans

~~Q. $A-B/(C*D)^E$~~

Q. $A-B/(C*D^E)$
Do in Postfix

Sol → $A - B / (C * \underset{\rightarrow M}{\underbrace{DE^}})$ (Let $DE^ = M$)

$A-B/(C*M)$

$A-B/\underset{\rightarrow N}{\underbrace{CM*}}$ (Let $CM* = N$)

Good Write

DATE : __/__/____
PAGE : _____

⇒ $A - \frac{B}{N}$

⇒ $A - \underbrace{BN/}_{\to O}$ (Let $BN/ = O$)

⇒ $A - O \Rightarrow AO-$

⇒ $ABN/-$

⇒ $ABCM*/-$

⇒ $ABCDE\hat{}*/-$ Ans

In Prefix

$A-B/(C*D\hat{}E)$

⇒ $A - B/(C * \underbrace{\hat{}DE}_{\to M})$ (Let $M = \hat{}DE$)

⇒ $A-B/(C*M)$

⇒ $A-B/\underbrace{*CM}_{\to N}$ (Let $N = *CM$)

⇒ $A-B/N$

⇒ $A - \underbrace{/BN}_{\to L}$ (Let $L = /BN$)

⇒ $A-L \Rightarrow -AL$

⇒ $-A/BN \Rightarrow -A/B*CM \Rightarrow -A/B*C\hat{}DE$ Ans

Q Convert in Prefix

$A/B\hat{}C + D$

Sol→ $\underbrace{A}_{\hat{}BC} + D$ (Let $N = \hat{}BC$)

$A/N + D$

$\underbrace{/AN}_{\to K} + D$ (Let $K = /AN$)

$+KD \Rightarrow +/AND \Rightarrow +/A\hat{}BCD$ Ans

Good Write

(Infix to Postfix)

Algorithm Method

DATE: __/__/____
PAGE: _____

Q ((P+Q)XS)^(T-U) Convert in postfix

Sol →

Symbol scanned	Stack	Expression
(	(	
(	((	
P	((	P
+	((+	P
Q	((+	PQ
)	(	PQ+
X	(X	PQ+
S	(X	PQ+S
)		PQ+SX
^	^	PQ+SX
(	^(	PQ+SX
T	^(	PQ+SXT
-	^(-	PQ+SXT
U	^(-	PQ+SXTU
)	^	PQ+SXTU-

PQ+SXTU-^ Ans

Good Write

DATE: __/__/____
PAGE: _____

Q. K+L-MXN+(O^P)X W/U/V XT+Q

Convert in Postfix.

Sol →

Symbol Scanned	Stack	Expression
K		K
+	+	K
L	+	KL
-	-	KL+
M	-	KL+M
X	-X	KL+M
N	-X	KL+MN
+	+	KL+MNX-
(	+(	KL+MNX-
O	+(	KL+MNX-O
^	+(^	KL+MNX-O
P	+(^	KL+MNX-OP
)	+	KL+MNX-OP^
X	+X	KL+MNX-OP^
W	+X	KL+MNX-OP^W
/	+/	KL+MNX-OP^WX
U	+/	KL+MNX-OP^WXU
/	+/	KL+MNX-OP^WXU/
V	+/	KL+MNX-OP^WXU/V
X	+X	KL+MNX-OP^WXU/V/
T	+X	KL+MNX-OP^WXU/V/T
+	+	KL+MNX-OP^WXU/V/TX+
Q	+	KL+MNX-OP^WXU/V/TX+Q

(ek hatao) some precedence

High precedence (some rakho)

Low precedence (sabhi uthalo) each by each compare

Ans → KL+MNX-OP^WXU/V/TX+Q+

Good Write

Q: (A + (B X C - (D / E / F) X G) X H)

Convert into Postfix

Sol:

Symbol Scanned	Stack	Expression
(	(	
A	(	A
+	(+	A
(	(+ (	A
B	(+ (	AB
X	(+ (X	AB
C	(+ (X	ABC
-	(+ (-	ABC X
(	(+ (- (	ABC X
D	(+ (- (	ABC X D
/	(+ (- (/	ABC X D
E	(+ (- (/	ABC X DE
/	(+ (- (/	ABC X DE /
F	(+ (- (/	ABC X DE / F
)	(+ (-	ABC X DE / F /
X	(+ (- X	ABC X DE / F /
G	(+ (- X	ABC X DE / F / G
)	(+	ABC X DE / F / G X -
X	(+ X	ABC X DE / F / G X -
H	(+ X	ABC X DE / F / G X - H
)		ABC X DE / F / G X - H X + Ans

Rule update → same precedence ka case ho toh stack mein rahega.

DATE: __/__/____
PAGE: _____

Infix to prefix

Q. N + (O ^ P) * W / U / V * T

Sol →

Reverse

T * V / U / W *) P ^ O (+ N

Symbol Scanned	Stack	Expression
T		T
*	*	
V		TV
/	*/	TV
U	*/	TVU
/	*//	TVU
*	*// *	TVUW
W	*// *	TVUW
)	*// *)	TVUW
P	*// *)	TVUWP
^	*// *)^	TVUWP
O	*// *)^	TVUWPO
(	*// *	TVUWPO^
+	+	TVUWPO^ *// *
N	+	TVUWPO^ *// *N

TVUWPO^ *// *N+

Reverse → +N*//*^OPWUVT Ans

Good Write

DATE: __/__/____
PAGE: _____

Q. K+L −M*N + (O^P) * W/U/V * T + Q.

Sol: Reverse → Q+T * V/U/W *)P^O(+ N*M − L+K

Symbol Scanned	Stack	Expression
Q		Q
+	+	Q
T	+	Q T
*	+*	Q T
V	+*	Q T V
/	+*/	Q T V
U	+*/	Q T V U
/	+*//	Q T V U
W	+*//	Q T V U W
*	+*//*	Q T V U W
)	+*//*)	Q T V U W
P	+*//*)	Q T V U W P
^	+*//*)^	Q T V U W P
O	+*//*)^	Q T V U W P O
(	+*//*	Q T V U W P O ^
+	++	Q T V U W P O ^ * // *
N	++	Q T V U W P O ^ * // * N
*	++*	Q T V U W P O ^ * // * N
M	++*	Q T V U W P O ^ * // * N M
−	++−	Q T V U W P O ^ * // * N M *
L	++−	Q T V U W P O ^ * // * N M * L
+	++−+	Z
K	++−+	Z K

Z K + − + +

++

Z K + − + +

Good Write

DATE: __/__/____
PAGE: _____

Q T V U W P O ^ * // * + N M − L k + −

⇒ Ans → − + k L − M N + * // * ^ O P W U V T Q

Ans → + + − + k L * M N * // * ^ O P W U V T Q

Postfix to Infix

Expression	Stack
a b c − + d e − f g − h + / *	a
− + d e − f g − h + / *	b − c a
+ d e − f g − h + / *	a + b − c
d e − f g − h + / *	e d a + b − c
f g − h + / *	d − e a + b − c
− h + / *	g f d − e a + b − c
h + / *	f − g d − e a + b − c
+ / *	h f − g d − e a + b − c
/ *	f − g + h d − e a + b − c
*	(d − e) / (f − g + h) a + b − c

Good Write $(a+b-c) * \left(\frac{(d-e)}{(f-g+h)}\right)$

University Ques

DATE: __/__/____
PAGE: _____

Q. 12, 7, 3, -, /, 2, 1, 5, +, *, +

Sol→

Expression	Stack
-, /, 2, 1, 5, +, *, +	3 7 12
/, 2, 1, 5, +, *, +	7-3 12
2, 1, 5, +, *, +	12/4 ⇒ 3
+, *, +	5 1 2 3 5+1 ⇒ 6
*, +	2 3 6*2 = 12
+	3

12+3 = 15 Ans

Q. 5, 6, 2, +, *, 12, 4, /, -

Sol→

Expression	Stack
⇒ +, *, 12, 4, /, -	2 6 5
⇒ *, 12, 4, /, -	8 5
⇒ 12, 4, /, -	40
⇒ /, -	4 12 40
-	12/4 ⇒ 3 40

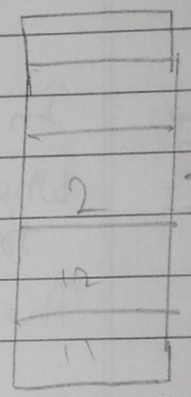
40 - 3 ⇒ 37 Ans

■ Topic → Stack

DATE: 13/9/23
PAGE:

Push Operation using array

```
void push()
{
int stack[5], item, top;
 if (top<4)
{printf("Enter stack item: "); scanf ("%d", &item);
 top= top + 1;
 stack[top] = item;
}
else
 printf("Stack is full");
}
```



POP using array

```
void pop()
{ int stack [5], item, top;
  if (top != -1)
   { item = stack [top];
     top= top-1;
     printf ("%d is removed", item);
   }
else
  printf ("Stack is empty");
}
```

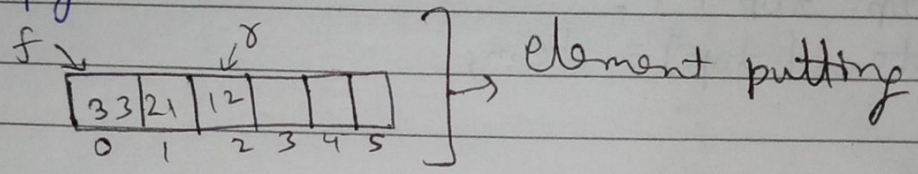
Topic → Queue

DATE: __/__/____
PAGE: _____

Based on FIFO :-

Major operation :
1. enqueue
2. dequeue

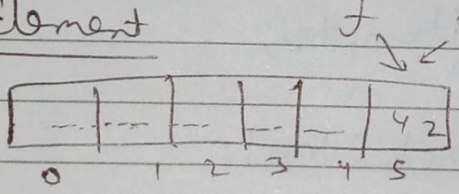
minor operation :
3. peek → front position element representation.
4. is full
5. is empty



Note: In empty queue ; rear = -1 & front = -1

Note: Whenever we put item firstly in (empty) queue ; then front & rear increased.
Then further only rear (index) value will increase.

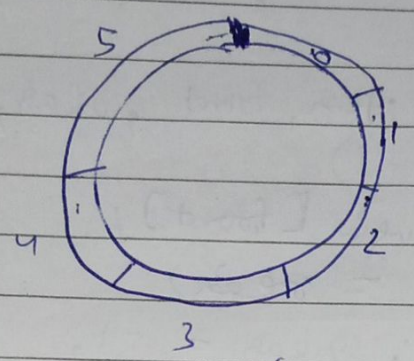
→ Deletion of Element



Limitation of simple Queue

1. If front & rear reach at last of Queue by one by one deleting queue element then no more item can be added in queue.

Hence Circular Queue comes into existence.
Ring



Here; rear = (rear + 1) % max size

rear = (5 + 1) % 6 = 0

front = (front + 1) % max size

Code

⇒ Put element in queue

```
void queue()
{ int queue[5], item, front, rear;
  if (rear < 4) {
  printf ("Enter value"); scanf("%d", &item);
  if (front = -1)
  { front = (front + 1);
   rear = rear + 1;
  }
  else
   { rear = rear + 1 ; }
  queue[rear] = item;
} }
```

Good Write

DATE: __/__/____
PAGE: _____

Delete from queue

```
void delete()
{
  int queue[5], item, front, rear;
  if (front != -1)
  { item = queue[front];
    if (front == rear)
    { front = -1; rear = -1;
    }
    else
    { front = front + 1; }
  }
  else
  { printf("Queue is Empty");
  }
}
```

⇒ Circular Queue insertion

```
void insert()
{  int queue[5], item, front, rear;

  if (front == (rear+1) % max_size)
   { printf("Queue full"); }
  else if (front == -1)
   { front += 1; rear += 1; queue[rear] = item; }
  else
   { rear = (rear + 1) % max_size;
   queue[rear] = item; }
  }
```

DATE: __/__/____
PAGE: _____

⇒ Circular Queue Deletion

```
void delete()
{ int front, rear, item, queue[10], maxsize = 10;
  if (front == -1)
  { printf ("queue empty");}
  else
  {item = queue [ front ] ;}

  if (front == rear)
   { front = -1;
     rear = -1;}
     else
    { front = (front + 1) % maxsize ;
     }
   }
```

To determine the address of a particular element in array

(1-D Array)

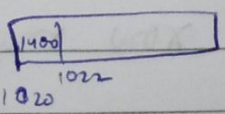
Q Given the base address of array B[1400 ___ 1800] is 1020 & size of each element is 2 bytes in the memory. Determine the address of B[1700].

Ans →

$$B + W(I - LB)$$

B → Base Address
W → size
I → जिसका find karna है
LB → Lower value

$$= 1020 + 2 \times (1700 - 1400)$$



Good Write

DATE: __/__/____
PAGE: ____

Q. The base address of theory A[-10 ..... 2] is 999. The size of each element is 2 bytes. Calculate the address of A[-1].

Sol⇒ B + w(I - LB)

⇒ 999 + 2[-1 - (-10)]

⇒ 999 + 2(10 - 1)

⇒ 999 + 18 ⇒ 1017

2-D array

Row Major

$B + W[N(I - LR) + (J - LC)]$

- B → base address
- W → storage size
- N → no. of Column
- LR → Low value of row
- I → Row subscript of element whose address is to be found
- J → Column subscript of element whose address is to be found
- LC → Lower bound of Column

Column Major

$B + W[(I - LR) + M(J - LC)]$

- I → Row subscript of element whose address is to be found
- M → No. of Rows in matrix
- J → Column subscript of element whose address is to be found
- LC → Lower bound of Column

$N = [UB - LB] + 1$

Q. Given the array A[4 ..... 7, -1 ..... 3] requires 2 byte of memory. The starting location is 100. Determine location A[6][2] using row major, column major implementation.

Good Write

Sol→ Row major A[6][2] I = 6 J = 2

$B + W[N(I-LR) + (J-LC)]$

$\rightarrow 100 + 2[5(6-4) + (2-(-1))]$

Column Major

$M = 7 - 4 + 1 = 4$

$B + W[(I-LR) + M(J-LC)]$

$100 + 2[(6-4) + 4(2+1)]$

$100 + 2[2+12]$

$\rightarrow 128$

Q: Give them array A[-20, --- 20, --- 10, --- 35] requires 1 byte of memory. The storing location is 500, determine the location of A[0][30] using row major, column major implementation.

Ans→ Row major

I = 0 J = 30

$B + W[N(I-LR) + (J-LC)]$

$500 + 1[26(0+20) + (30-10)]$

35-10+1

$500 + [26 \times 20 + 20]$

26

$500 + 20(27) = 500 + 540 = 1040$

Column Major

$500 + 1[(0+20) + 41(30-10)]$

$500 + [20 + 41 \times 20]$

20+20+1

$500 + 20 + 820 = 1340$

41

PAGE :

Q. Given the array S[-15 ..... 10, 15, ... 40] require 1 byte of the storage. The starting location is 1500. Calculate the address S[15,20] using row & column Major implementation

Sol:

Row major

I = 15 J = 20

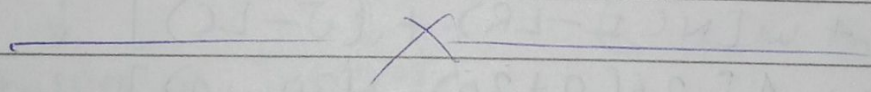
40-15+1
26

1500 + 1 [26(15+15) + (20-15)]
1500 + (26×30) + 5 ⇒ 2285

Column

1500 + 1[(15+15) + 26(20-15))
1500 + 30 + (26×5)
1500 + 30 + 130 ⇒ 1660

10+15+1
26



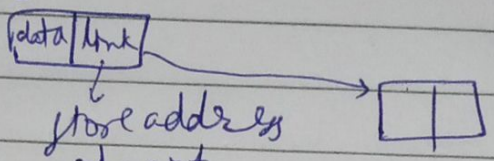
Good Write

DATE: 18/9/23
PAGE:

Topic → Linked List

→ It is linear data structure.
→ It is collection of nodes.
→ No contigious allocation.

node has 2 parts



Merit

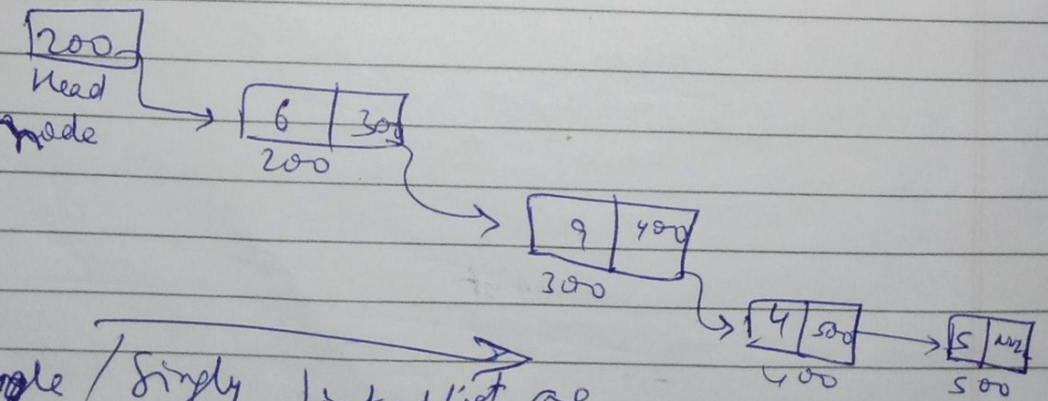
→ Prevent memory wastage.
→ Possible Dynamic memory allocation.
→ Insertion & deletion is simpler as compared to array.

Limitation

→ Direct access not possible as possible in array.
Sequential access is there.

Implementation

```
struct node
{ int data;
  struct node * next;
}
```



Single / Singly Linked List as direction is only in one side

Good Write

■ Sub-topic → Singly Linked List

DATE: __/__/____
PAGE: _____

Insert elements (instart)

Main hint:

ptr.next → Head
head = ptr

Code:

```
struct node
{ int data;
  struct node * next;
  }; struct node * head;
void insert() {
struct node * ptr;
int item;
ptr = (struct node *) malloc ( size of (struct node *)) ;
if (ptr == NULL)
  { printf ("No space");}
else
{ printf ("Enter value : "); scanf ("%d", & item);
// ptr.data → item ;  ptr → data = item;
// ptr.next → head;  ptr → next = head;
  head = ptr ;
}}
```

[Insert elements (in end)]

Code

```
struct node
{ int data ;
struct node * next ;
};
struct node * head;
```

```
void insert (){
  struct node *ptr, *temp;
  int item;
  ptr = (struct node*) (sizeof (struct node*));
if (ptr == NULL) printf("overflow"); else {
  temp = head;
while ( temp -> next != NULL)
  { temp = temp -> next;
    }

/*temp.next -> ptr; */ temp -> next = ptr;
  printf("Enter value");
  scanf("%d", &item);
/* ptr.next -> NULL; */ ptr -> next = NULL;
/*ptr.data -> item; } */ ptr -> data = item;
  }
```

DATE: 19/9/23
PAGE:

Random insert

Code

```
struct node
{ int data;
  struct node * next ;
};
struct node *head;
void Rinsert()
{ int item,loc; int i;
  struct node * ptr ; *temp; *ptr1 ;
  ptr = (struct node *) malloc (sizeof (struct node*));
  If (ptr == NULL)
  { printf ("Error");
  }
  else
  { printf ("Enter data:");
    scanf ("%d", & item);
    ptr → data = item ;
  temp = head ;
  printf ("Enter position before which you want
    to insert new node");
  scanf ("%d", & loc);
  for (i = 1 ; i < loc ; i++)
  { ptr1 = temp;
    temp = temp → next ; }
  ptr → next = ptr1 → next ;
    ptr1 → next = ptr ;
  } }
```

Common

DATE: __/__/____
PAGE: _____

Deletion

At beginning

Common

```
temp = head ;
head = temp -> next;
free (temp);
```

At last

```
struct node
{ int data;
  struct node* next;
};
struct node *head;
void ldelete()
{ struct node * temp, * ptr1;
   temp= head;
  while ( temp-> next != NULL)
  { ptr1 = temp;
    temp= temp->next;
    }
  ptr1->next = NULL;
   free (temp);
}
```

Good Write

DATE: __/__/____
PAGE: _____

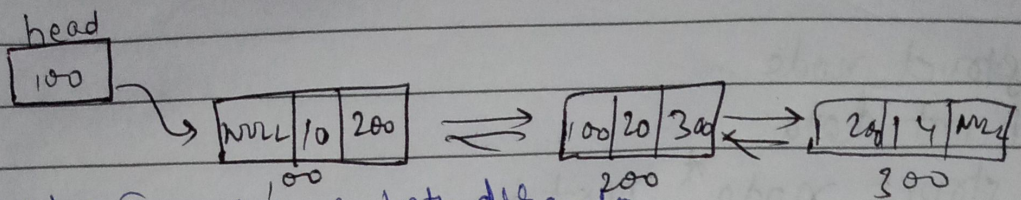
Random

```
struct node
{ int data;
 struct node * next;
 };
struct node * head;
void rdelete ( ){
 struct node *temp, * ptr1;
 int loc, int i;
 printf ("Enter loc to del: "); scanf ("%d", &loc);
temp=head;
 for(i=0; i< loc; i++)
{ ptr1 = temp;
   temp = temp→ next;
 }
ptr1 → next = temp → next;
   free (temp);
}
```

Good Write

DATE: 21/9/23
PAGE:

Sub-topic → Doubly Linked List



Merit → Traversing in both direction.

Demerit: 1. Additional memory occupied. 2. More complexity (due to both direction traversing)

Insertion

At beginning

```
struct node
{ int data;
  struct node * next;
  struct node * prev;
};

struct node *head;
void insert() {
struct node *ptr;
int item;

ptr = (struct node*) malloc(sizeof(struct node *))
if (ptr == NULL)
{ printf("Error"); }
else
{ printf("Enter data:"); scanf("%d", &item);
  ptr->data = item;
    ptr->prev = NULL;
    head->prev = ptr;
      ptr->next = head;
    head = ptr;
  }
}
```

DATE: __/__/____
PAGE: _____

At end

Common

```
struct node
{ int data;
  struct node * next;
  struct node * prev;
};
struct node * head;
void Insert ()
{ struct node * ptr, * temp;
  int item;
  ptr = (struct node *) malloc (sizeof (struct node *));

  if (ptr == NULL)
  { printf ("Error");}
  else
  { printf ("Enter data:"); scanf ("%d", & item);
    ptr -> data = item;
    temp = head;
    while (temp -> next != NULL)
    { temp = temp -> next;
    }
    temp -> next = ptr;
    ptr -> prev = temp;
    ptr -> next = NULL;
  }
}
```

Random after loc

Common

```
void insert () {
struct node *temp, *ptr, *ptr1;
int item, loc;

if (ptr == NULL)
  printf("Error");
 else
 { printf("Enter data & loc:");
  scanf ("%d %d", &item, &loc);
  ptr->data = item
  temp = head;
  for (i=1; i<=loc; i++)
    { ptr1 = temp;
      temp = temp->next ;
    }

  ptr1->next = ptr;
    ptr->prev = temp->prev;
     ptr->next = temp;
    temp->prev = ptr;
 }
}
```

DATE: __/__/____
PAGE: _____

Deletion

Start

Common

```
void sdelete ()
{ struct node *temp;
  temp = head;
  head = head -> next;
  temp -> next -> prev = NULL;
    free (temp);
}
```

Last

Common

```
void ldelete ()
{ struct node *temp;
  temp = head;
    while ( temp -> next != NULL)
    { temp = temp -> next; }
    temp -> prev -> next = NULL;
      free (temp);
}
```

DATE: __/__/____
PAGE: _____

Random

common

```
void delete() {
struct node *temp, *ptr1;
int loc,int i;
printf("Enter loc:"); scanf("%d",&loc);
 temp=head;
  for(i=1;i<loc;i++) {
   ptr1=temp;
   temp = temp->next;
  }
  ptr1 -> next = temp -> next;
  temp->next->prev =ptr1;
   free(temp);
  }
```

Imp Ques

1. One ques on linked list.
   Infix Prefix Postfix

DATE: __/__/____
PAGE: _____

Note → Sparse Symmetric Matrix

Matrix having zeroes more than non-zero elements.

$$\begin{bmatrix} 0 & 1 & 3 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 1 & 4 & 0 \\ 2 & 0 & 0 & 1 \end{bmatrix}_{4\times4}$$

Regular

Irregular

Lower Triangular

Upper Triangular

Tri Diagonal

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 6 & 2 & 4 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 2 & 2 \\ 0 & 3 & 2 & 6 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 5 \end{bmatrix} \quad \begin{bmatrix} 1 & 5 & 0 & 0 \\ 2 & 6 & 4 & 0 \\ 0 & 7 & 6 & 2 \\ 0 & 0 & 6 & 2 \end{bmatrix}$$

Row	Column	Value
0	0	2
0	2	1
1	1	1
1	3	2
2	0	3
2	1	1
3	2	1

Note: Don't represent 0 in memory

Good Write

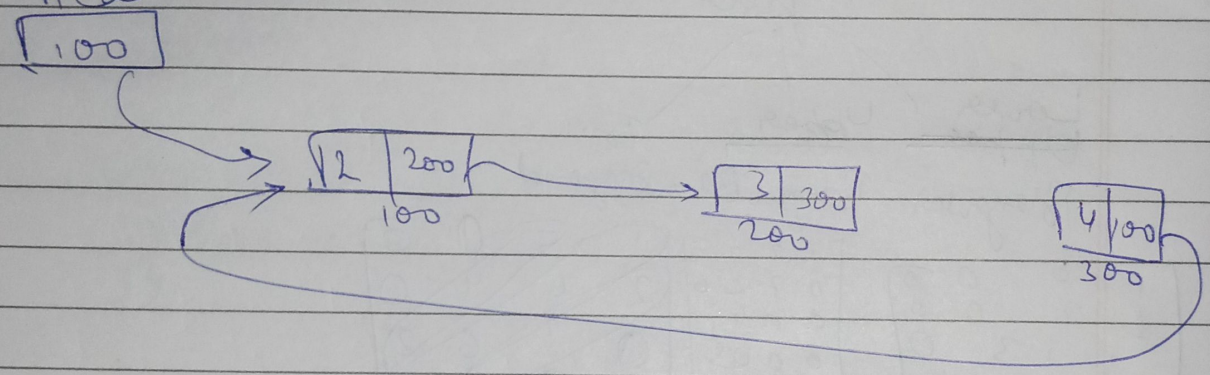
DATE: __/__/____
PAGE: _____

Merit
~~Not given~~

- To have a effecient memory utilization becoz in sparse symmetric matrix zero doesnot occupy any space.

Circular Linked List

It is extension of singly linked list.
Merit: To tackle demerit of doubly linked list.



Code

Insertion

At beginning

```
struct node
{ int data;
  struct node *next; };
struct node* head;
void insert( ){
  int item;
  struct node *temp, *ptr;
  ptr = (struct node*) malloc (sizeof (struct node*));
    if (ptr == NULL)
      printf ("Error");
```

```
else
{ printf("Enter item:"); scanf("%d",&item);
   temp ptr->data = item;
    temp= head;
    while (temp->next != head)
    { temp= temp->next;
    }
    temp->next = ptr;
    ptr->next = head;
      head = ptr;
    }}
```

At last

Common

```
struct node
{ int data;
  struct node * next;
};
struct node * head;
void Insert ( ){
 int item;
struct node *temp, *ptr;
ptr = (struct node *) malloc (sizeof (struct node*));
if (ptr == NULL) {printf("Error");}
else
{ printf("Enter item"); scanf("%d", &item);
   ptr->data = item;
   temp = head;
  while ( temp->next != head)
   { temp = temp->next;
   }
   temp->next = ptr;
    ptr->next = head;
   }}
```

DATE: __/__/____
PAGE: _____

Random

Code

```
struct node
{int data; struct node * next; };
struct node * head;
void ginsert() {
int item, loc; int i;
struct node * ptr, * temp, * ptr1;
 ptr = (struct node *) malloc (sizeof (struct node *));
 if (ptr == NULL)
 { printf ("Error"); }
else
{ printf ("Enter data:"); scanf ("%d", &item);
  printf ("Enter pos at which you want to insert
  element: "); scanf ("%d", &loc);
  ptr->data = item;
  temp= head;
  for (i=1; i<loc ; i++)
  { ptr1= temp;
    temp = temp-> next;
  }
  ptr-> next = ptr1 -> next;
  ptr1-> next = ptr ;
  } }
```

Deletion

From Beginning

Common

```
void sdelete ( ) {
struct node *temp , *ptr1, *temp 2;
temp=head ; ptr1 = head ; temp2 = ptr1 → next;
 while ( temp→ next ! = head)
   { temp= temp → next;
     }
  temp → next = temp2;
   head = temp2 ;
    free ( ptr1);
   }
```

Another way

```
struct node *temp,
 * ptr1;
temp=head;
while (temp→next!=head)
{temp= temp → next;
}ptr1 = head ;
temp→ next = head→ next;
head = temp → next;
free (temp);
free ( ptr1);
```

From last

Common

```
void ldelete ( )
{ struct node *temp, *ptr1;
   temp=head ;
   while ( temp→ next ! = head)
     ptr1 = temp;
     { temp= temp→ next; }
     ptr1 → next = head;
      free ( temp);
      }
```

DATE : __/__/____
PAGE : _____

Random

```
struct node
{ int data; struct node *next;};
  struct node * head;
void delete(){
int loc, i;
struct node * temp, *ptr1;
printf("Enter loc to delete :"); scanf("%d", &loc);
  temp=head;
for(i=1; i<loc; i++)
{ ptr1= temp;
temp= temp->next;}
ptr1->next = temp->next;
   free(temp);
}
```