2/9/18

ARRAYS

→ collection of data elements of the same data type

ARRAYS

- NUMERIC ARRAYS
  - 1D
  - 2D
- CHARACTER ARRAYS — called as strings

1D (row)

10	20	30

10
20
30

(column)

→ 1 loop for every step

2D

10	20	30
20	45	50
60	75	90

(rows & columns)

→ 2 loops for every step

1D ARRAY

DECLARATION

0	1	2	3

(DATA TYPE) (space) (NAME OF ARRAY) [size of array];

square brackets

eg: int A[5]; (different, as 0→4 = 5)

0	1	2	3	4
10	20	30	40	50

index no. / subscript values

→ always start from 0 → valid location in C++

→ shows position / location of an element in an array

SOME STUFF:

→ cout << A[2]; // 30

cout << A[2] += 10 // 40

STATIC ~~Starting~~ data structure
* Derived data type

ARRAYS ARE ~~STARTING~~ STATIC DATA STRUCTURES

Collection of elements

This is because:

→ The no. of elements in an array is pre-defined and cannot be changed during the course of a program.

eg: int A[10]; ✓ (ONLY FOR DECLARATION)

int n;
int A[n]; X

const int n = 10;
int A[n]; ✓ (as n is a constant)

* ~~Write~~ Create an array called "fun" having 30 elements of data type float.

→ ns. float fun[30];

DATA USAGE OF AN ARRAY

→ data used = (data type data used) * (total index values)

→ eg: float fun[30];

data used = 4 * 30
= 120 bytes

* ~~Write~~ WAP to input an array "sum" having a max. 20 elements & display the array with each element on a separate line.

Ans.

```
#include <iostream.h>
void main(){
    int sum[20];
    int i;
    for(i=0; i<20; ++i){
        cin >> sum[i];          // input
    }
    for(i=0; i<20; ++i){
        cout << sum[i];         // display
    }
}
```

* Input an array 'masks' for 'n' students & display those masks which is greater than 60.

```
#include <iostream.h>
void main(){
    float masks[20];
    int i, n;
    cin >> n;
    for(int i=0; i<n; ++i)
    {
        if(a[i]>60)
            cout << a[i];
    }
```

sum

* Input an int array of 'n' elements and find sum of all even data elements of the arrays.

Ans.
```
#include <iostream.h>
void main(){
  int elements[20];
  int n, i, sum=0;
  cin>>n;

  for(i=0; i<n; ++i){
    cin>>elements[i];          // input
  }

  for(i=0; i<n; ++i){
    if(elements[i]%2==0){
      sum+=elements[i];        // odd/even
    }
  }

  cout<<sum;                   // display
}
```

* Input an int array of 'n' elements & count the no. of positive and neg. elements.

```
int array[20];
int i, n, nop=0, non=0;
cin>>n;

for(i=0; i<n; ++i){
  cin>>array[i];
}
```

Ans.

```
for (i=0, i<n, ++i){
    if (array[i] < 0){
        non++;                    // neg.
    }
    if (array[i] > 0){
        nop++;                    // pos.
    }
}
cout << non << endl << nop;
```

* create an array 'temp' which takes in the weekly temperature of a city, and hence calculate the avg. temp.

```
int temp[20];
int i, n;
float sum, avg;
cin >> n;
for (i=0; i<n; ++i){
    cin >> temp[i];               // input temp
}
for (i=0; i<n; ++i){
    sum += temp[i];               // sum
}
cout << sum/n;                    // output avg.
```

* Input an int array of 'n' elements, and search for a given data D as input by the user.

```
int A[10];
int n, i, D, flag=0; //user input

cin>>n>>D;

for(i=0; i<n; ++i){
    cin>>A[i];
}

for (i=0; i<n; ++i){
    if (A[i]==D){
        flag=1;
        break;
    }
}

if (flag==1){
    cout<<"found"<<D;
}
else cout<<D<<"not found";
```

Ans.

also called as 'divide & conquer'

SEARCH

Linear search
→ pref. small
→ unsorted / sorted

START
→ When started

STOP
→ When element found
→ When element is NOT found till the end

Binary Search
→ pref. large list
→ sorted only (pre-requisite)

START
→ When started

STOP
→ When F<=L

```
int A[20];
int B=0, n;
cin>> n;
```

LINEAR SEARCH

*① Create a 1D array of int & modify the array by multiplying all even elements by 2 and all odd elements by 5 & display.

*② Input a 1D array of int & display the array in the reverse form (order).

*③ Input a 1D array of int & find the sum of all the nos. whose last digit is 7.

①
```
int A[20];
int i; //d;
for(i=0; i<20; ++i){
    A[i] = i;          // create
}

for(i=0; i<20; ++i){
    if(a[i]%2==0){
        a[i]*=2;       // even *2
    }
    else{ a[i]*=5;
    }
}
for(i=0; i<20; ++i){
    cout<<a[i];
}
```

②
```
int A[20];
int i, n;
cin>>n;

for(i=0; i<n; ++i){
    cin>>A[i];         // input
}
```

Ans.

```
for(i=n-1; i>=0; --i){ // reverse order
    cout << a[i];      // display
}
```

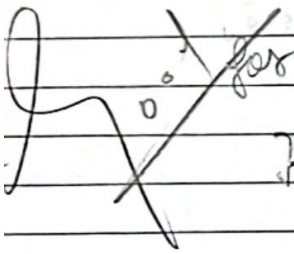
(3)
```
int A[20];
int i, n, ld, sum = 0;
cin >> n;

for(i=0; i<n; ++i){
    cin >> A[i];          // input
}
for(i=0; i<n; ++i){
    ld = A[i] % 10;
    if(ld==7){ sum += A[i]; } // ld
}
cout << sum;
```

(4) (i) Input 2 1D arrays having 'n' no. of elements in them.

(ii) Find the sum of the elements of the arrays, and store in a 3rd array.

(on next page)

```
int a[20], b[20], c[20];
int i, n;
cin >> n;

for(i=0; i<n; ++i){
    cin >> a[i] >> b[i];   // input
}

for(i=0; i<n; ++i){
    c[i] = a[i] + b[i];   // add
}

for(i=0; i<n; ++i){
    cout << a[i] << b[i] << c[i];   // display
}
```



Find the largest element in an array.

```
int a[20];
int i, n; lar = a[0];
cin >> n;

// input array

for(i=0; i<n; ++i)
{   if(a[i] > lar)
    {
        lar = a[i];
    }
}

cout << lar;
```

BINARY SEARCH

I. UNDERSTANDING (NO CODING)

0	1	2	3	4	5	6	7	8	9
5	8	11	15	17	23	25	32	48	63

1) Search for (25)

→ AF = 0
L = 9
M = (F+L)/2 = (0+9)/2
= 4.5 = 4 // ant

→ A[M] = A[4] = 17
17 < 25
A[M] < data
→ ∴, F = M+1 = 4+1
F = 5
L = 9
M = (5+9)/2 =
M = 7
A[M] = A[7]
= 32
32 > 25
A[M] > data
→ ∴, F = 5
L = M-1 = 7-1
M = 6
M = (5+6)/2 = 5.5
= 5

A[M] = A[5]
= 23
23 < 25
A[M] < data
∴ F = M+1 = 5+1
= 6
L = 6
M = (6+6)/2
= 6
A[M] = A[6]
= 25
25 = 25
∴ index value of 25 = 6.

2)

3	6	9	11	13	18	23	32	46	51	65	73

USING BIN SRCH

* Input an array of 'n' elements, and perform binary search.

```
int array[20];
int i, n, data, f, l, m, flag=0;
cin>>n;

for(i=0; i<n; ++i){
    cin>>n;
}
cin>>data;

f=0;
l= n-1;          // index value starts from 0

while(f<=l){          //
    M=(f+l)/2;

    if(A[M]==data){
        flag=1;
        break;
    }
    else if(A[M]>data){          //
        l=M-1;                   //
    }

    else{
        f=M+1;                   //
    }
}                                //end of loop

if(flag=1){cout<<"found @"<<M<<"th pos.";}
    else{cout<<"not found"}
```

no. of elements = n
no of passes = n-1
passes need = ____

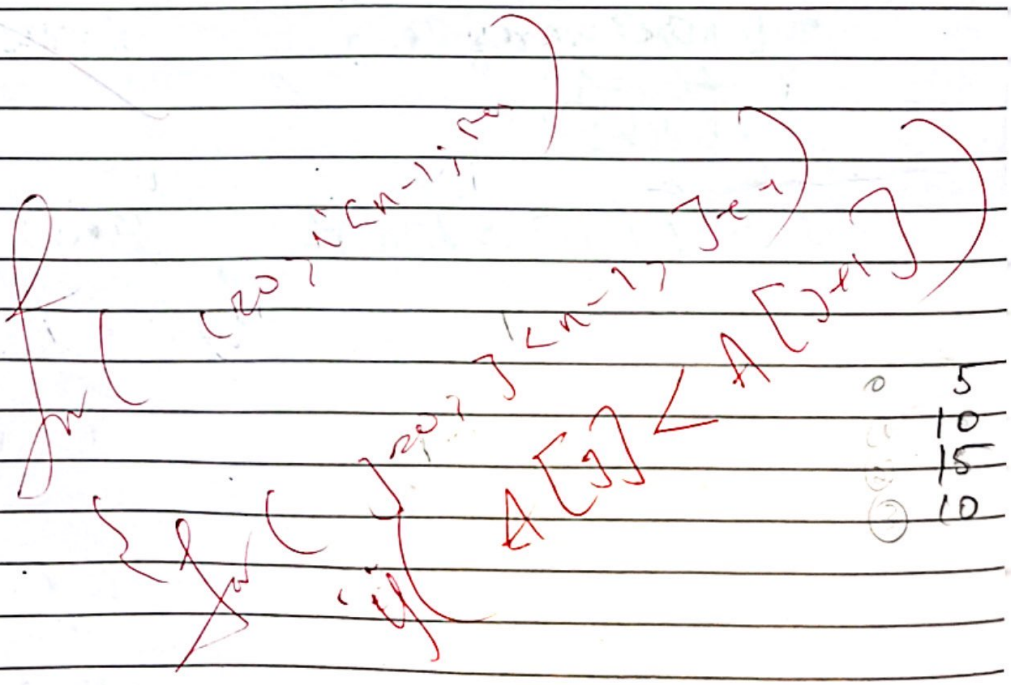
SORTING

BUBBLE SORTING
→ less memory
→ more processing time
→ multiple interchanges can be made in 1 pass

SELECTION ...
→ more memory
→ less processing time
→ only upto 1 interchange can be made in 1 pass

INSERTION (GRADE 1

ARGER NO. IN
BUBBLE
EEP INTER HANGING

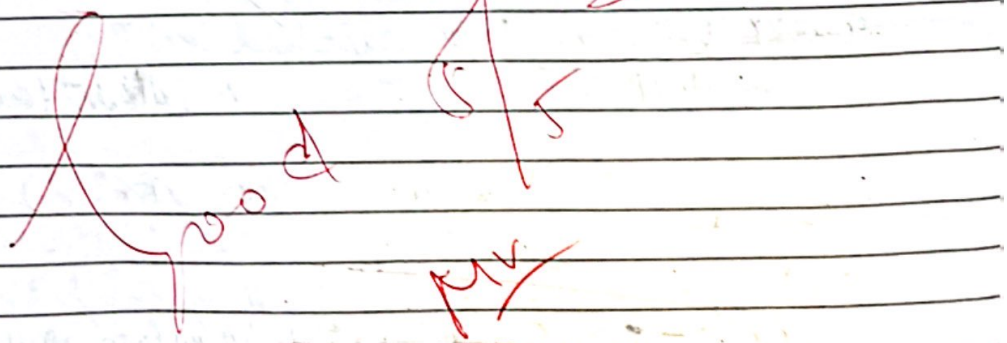


ASSIGNMENT — 12th

* Input an array of 'n' nos. Sort it out (in descending order) & perform binary search to search for a given element & give appropriate messages.

Ans.
```
#include <iostream.h>
void main(){
    int array[20];                          //array
    int i, n, j, dummy, data, F, L, M, flag;

    cout<<"Enter no. of elements"; cin>>n; //input n
//acquiring data from user
    cout<<"Enter your nos.";
    for(i=0; i<n; ++i){
        cout<<(i+1)<<"th no.:";
        cin>>array[i];
    }
    cout<<"This is what you've entered:";
    for(i=0; i<n; ++i){
        cout<<array[i];                     //display inputted
    }

    cout<<"Enter no. to be searched for:";
    cin>>data;                              //input search no.

//sorting data (in descending order)

    for(j=1; j<=n-1; ++j){                  // no. of passes
        for(i=n-1; i>0; --i){               // index value
            if(array[i-1]>array[i]){        //if 2nd>3rd
                dummy=array[i];
                array[i]=array[i-1];        //interchange
                array[i-1]=dummy;
            }
        }
    }
```

```
//searching

F=0;                          // 1st term
L=n-1;                        // Last term
flag=0;
while(F<=L){
    M=(F+L)/2;                // Middle term

    if(A[M]==data){
        flag=1; break;
    }
    else if(A[M]<data){
        L=M-1;
    else{
        F=M+1;
    }
}
if(flag==1){
    cout<<"found";
}
else{ cout<<"Not found";
}
//end of void main()
```

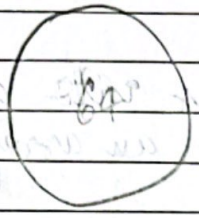


BUBBLE SORTING

I CONCEPT (NO CODING)

* Sort in ascending order:

	①	②
12	12	12
73	62	10
62	10	5
10	5	8
5	8	62
8	64	
64	7	
7	15	
15	73	



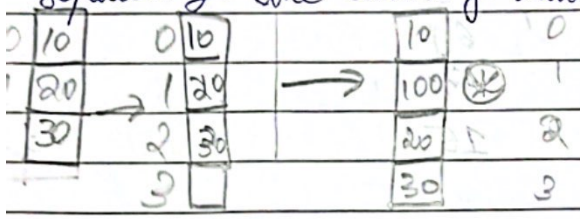
II CODING

```
int array[20];
int i, j, n, t;   // t is dummy var for 3 bucket
cin>>n;

for(i=0; i<n; ++i){
    cin>> a[i];          //input
}
for(i=0; i<n; ++i){
    cout<< a[i];         // output (simply)
}

for(j=1; j<=n-1; ++j)    //counter for no. of passes   (max no. of passes)
{
    for(i=0; i<n-1; ++i)  //counter for no. of element   (we only compare 1st & 2nd, 2nd & 3rd; NOT j & j+1)
    {
        if(A[i+1]<A[i])
        {
            t=A[i+1];          // 3 bucket system (3rd becomes 2nd; 2nd becomes 3rd)
            A[i+1]=A[i];
            A[i]=t;
```

```
for(i=0; i<n; ++i){
    cout<< A[i];          //display
}
```

INSERTION

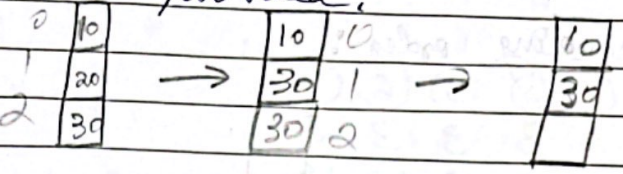
Insert an element given by user into a user-specified location, into an array without replacing the existing data




```
int a[20];
int i=0, n, data, pos, j=n-1;
cin>> n >> data >> pos;

for(i=0; i<n; ++i)
{   cin>> a[i];
}
while(j>=pos)
{   a[j+1] = a[j];
    j--;
}
a[pos] = data;
n++;
```

DELETION

* Inserted an array XXXXX & delete the element from the assay. Data to be deleted is user-provided.




```
int assay[20];
int i, n, flag=0, pos=0, j=n-1;
```

// input

```
for(i=0; i<n; ++i){                    // check & get pos.
    if(assay[i]==data){
        pos = i;
        flag = 1;
        break;
    }
}
int j=pos;
while(j<n)                             // so null char or something replaces last spot
{   a[j]=a[j+1];
    j++;
}
n=n-1;                                 // remove empty space @ end.
if(flag==0) cout<<"not found no.";
```

SELECTION SORTING

I CONCEPT

* Sort in ascending order:

	①	②	③	④	⑤	⑥	⑦
~~75~~	3	3	3	3	3	3	3
③	75	12	12	12	12	12	12
18	18	18	18	18	18	18	18
53	53	53	53	32	32	32	32
72	72	72	72	72	44	44	44
63	63	63	63	63	63	53	53
44	44	44	44	④	72	72	63
32	32	32	③②	53	⑤3	⑥3	72
12	①2	75	75	75	75	75	75

no. of elem. = 9

Max no. of passes = (9) - 1
= 8

no. of passes used = 7

* Sort in asc. order using both sorts:

BUBBLE

①	②
-2	-2
4	3
3	-1
-1	0
0	-5
-5	4
7	-6
-6	5
5	7

-2
4
7

II CODING

```
int a[20];
int i, n, j, m, pos;
cin>>n;

for(i=0; i<n; ++i){
    cin>>a[i];
}

for(i=0; i<n; ++i){
    cout<<a[i];
}

for(j=0; j<n-1; ++j)                    // j -> no. of passes
{   m=a[j];                             // j=0 because
    pos=j;                              // we need a[j]=a[0]

    for(i=j+1; i<n; ++i)                // i -> no. of element for comparison
    {   if(a[i]<m)                      // smallest term
        {   m=a[i];
            pos=i;
            break;                      // interchange comparison
        }                               // term & smallest term
    }                                   // for comparison
    a[pos]=a[j];
    a[j]=m;
}
```

don't use these

You might be thinking what happens if 'if' is never true...

Then, a[j] is the smallest term, and will remain there.

Switch elements within each pair

10	15	20	25	30	35

↓

15	10	25	20	35	30

```
for(i=0; i<n; i+=2){
    dummy = a[i];
    a[i] = a[i+1];
    a[i+1] = dummy;
}
```

10	20	30	40	50	60	70

↓

70	60	50	40	30	20	10

Reverse an array

```
int i=0, j=n-1, dummy;
while(i<j){
    dummy = a[i];
    a[i] = a[j];
    a[j] = dummy;
    ++i;
    --j;
}
```

10	20	30	40	50

↓

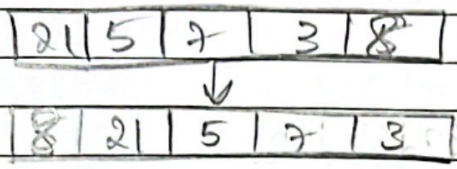
10	20	25	30	40	50

(inputted by user)

```
int i, j=n-1, pos;
for(i=0; i<n; ++i){
    if(a[i] >= data){
        pos = i;
        flag = 1;
        break;
    }
}
```

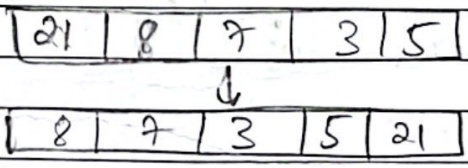
(*) // not req.

```
while (j >= pos) {
    a[j+1] = a[j];
    j--;
}
a[pos] = data;
n += 1; // as we're adding an element
```

* 

(move 'last element to first)

```
int dummy, j=n-1;
dummy = a[n-1]; //not j, as j changes
while(j>0){
    a[j] = a[j-1];
    j--;
}
a[0] = dummy;
```

* 


```
int dummy = a[0], i=0;
while (i<n-1)
{   a[i] = a[i+1];
    i++;
}
a[n-1] = dummy;
```

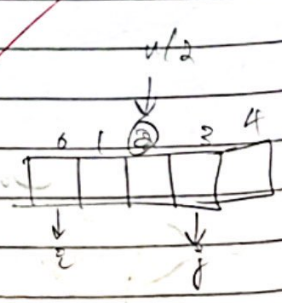
*

10	20	30	40	50	60

↓

40	50	60	10	20	30

```
cin>>n;
int i=0, dummy, j=n/2;
if(n%2==0) //even
while(i<n/2)
{
    dummy = a[i];
    a[i] = a[j+i];
    a[j+i] = dummy;

    i++;
    j++;
}
else //odd
{   i=0; j=(n/2)+1;

    while(i<n/2)
    { int t = a[i];
      a[i] = a[j];
      a[j] = t;
    }
}
```



0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3

2-DIMENSIONAL ARRAYS

① DECLARATION

ROWS COLUMNS

<data type> <space> <name> [ ] [ ]

② MAX NO. OF ELEMENTS

= Max no. of rows X Max no. of columns

③ ORDER

no. of rows X no. of columns

(5x3), (4x2), ...

```
for(i=0;i<r;++i){
  for(j=0;j<c;++j){
    // cin>>a[i][j];
    // cout<<a[i][j]<<endl;
  }
  cout<<endl;
}
```

* Create an array of 3 rows & 4 columns. Input the elements and display them.

100	200	300	400
50	80	30	20
60	100	50	100

Input and display an array of ~~r & c~~ r rows and c columns.

1) find sum & avg. of even nos.

```
int A[10][10];
int i,j, r, c, esum=0, enum=0;
float avg=0;                          // avg can be dec
cin>>r>>c;
for(i=0; i<r; ++i){
    for(j=0; j<c; ++j){
        cin>>A[i][j];                 //input
    }
}

for(i=0; i<r; ++i){
    for(j=0; j<c; ++j){
        cout<<a[i][j]<<endl;          //display
    }
    cout<<endl;
}

for(i=0; i<r; ++i){
    for(j=0; j<c; ++j){
        if(A[i][j]%2==0){             // if even
            esum+=A[i][j];
            enum++;
        }
    }
}

avg=esum/enum;                        // calc avg

cout<<esum<<endl<<enum;               // display
                                      & esum
```

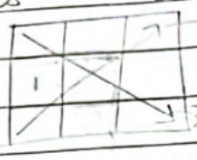
* Input a 2-D array for searching an inputted no. in the array.

Ans.
```
int a[10][10];
int i, j, r, c,
for(i=0; i<r; ++i){
    for(j=0; j<c; ++i){
```
Am >>

Please complete your code.

EQUALS OR SQUARE MATRIX

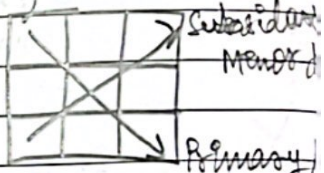
... is a 2-D array where no. of rows = no. of columns



Input a 2-D array, and display the diagonal elements.

```
int a[10][10],
Int i, j, n;
cin >> r;
// Input the array
for(i=0; i<n; ++i){
    for(j=0; j<n; ++j){
        if(i==j){
            cout<< a[i][j]; // Prim.
            cout<<endl;     // for separation
        }
        if(i+j==n-1){
            cout<<a[i][j]; // Sub.
        }
    }
}
```

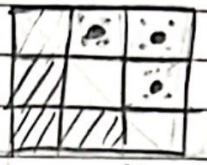
(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)
(2,0)	(2,1)	(2,2)



do only ①

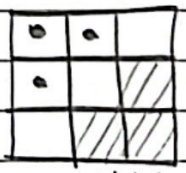
* Input a 2-D array and display upper & lower Δ.

```
// array
// int
cin >> r;

// input array
① Boundary diag
for(i=0; i<r; ++i){
    for(j=0; j<r; ++j){

        if(i<j){
            cout << a[i][j];        // upper
            cout << endl;
        }

        if(i>j){
            cout << a[i][j];        // lower
        }
    }
}
```



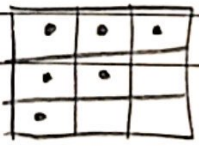
• upper
/// lower

```
② Subsidiary diag.
for(i=0; i<r; ++i){
    for(j=0; j<r; ++j){

        if(i+j<r-1){
            cout << a[i][j];
            cout << endl;
        }

        if(i+j>r-1){
            cout << a[i][j];
        }
    }
}
```

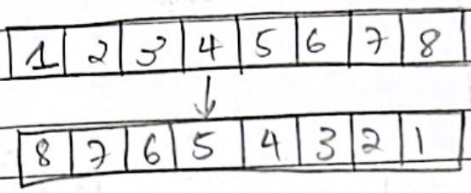


• upper
/// lower

not including middle, i.e.



CHALLENGE

① 


```
#include <iostream.h>
void main(){
  int a[20];
```

②

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)
(2,0)	(2,1)	(2,2)
(3,0)	(3,1)	(3,2)

find sum of each row;

```
for(i=0; i<R; i++){
    sum=0;

    for(j=0; j<C; ++j){
        sum+ = a[i][j];
    }
}
```

③

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

col sum

```
for (j=0; j<C; ++j){          // increment col
    csum=0;
    for(i=0; i<R; i++){       // increment rows
        csum+ = a[i][j];
    }
}
```

Places switched because we need to increment rows first

19/18 ASSIGNMENT

~~Find the sum~~

Display the middle row elements

```
#include <iostream.h>
void main()
{   int a[20][10];
    int i, j, r, c;
    cout<<"Enter no. of rows & columns";
    cin>>r>>c;
    cout<<"Enter your elements";
    for(i=0; i<r; ++i)
    {  for(j=0; j<c; ++j)
       {  cin>>a[i][j]; }                    //input
    }
    cout<<"Middle row elements";
    for(i=0; i<r; ++i)                       //Middle row
    {  for(j=0; j<c; ++j)
       {  if(i==r/2) cout<<a[i][j]; }
    }
    cout<<"Middle column elements:";
    for(i=0; i<r; ++i)
    {   for(j=0; j<c; ++j)                   //Middle column
        {
            if(j==c/2)
            {  cout<<a[i][j];
            }
        }
    }
}  //end of void main()
```

* Find the sum of middle - row elements

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

    cout<<"Enter no. of row & columns";
    cin>>r>>c;

    for(i=0; i<r; i++)                    //input
    {   for(j=0; j<c; ++j)
        {   cin>>a[i][j];
        }
    }

    for(i=0; rsum=0; i<r; i++)            // row sum
    {   for(j=0; j<c; j++)
        {   if(i==r/2)
            {   rsum+=a[i][j];
            }
        }
    }
    cout<<"The middle-row sum is"<<rsum;

    for(i=0; csum=0; i<r; i++)            //column sum
    {   for(j=0; j<c; j++)
        {   if(j==c/2)
            {   csum+=a[i][j];
            }
        }
    }
    cout<<"The middle-column sum is"<<csum;
}  //end of void main()
```

posr1=0, posr2=0;
posc1=0, posc2=0;

* Find smallest & largest element of an array display the position.

```
int a[10][10];
int r, c, min = a[0][0], max = a[0][0];
cin >> r >> c;
// input elements
for(i=0; i<r; ++i)
{   for(j=0; j<c; ++j)
    {
        if(a[i][j] < min)
        {   min = a[i][j];
            posr1 = i;
            posc1 = j;
        }
        if(a[i][j] > max)
        {   max = a[i][j];
            posr2 = i;
            posc2 = j;
        }
    }
}
cout << min << posr1 << posc1 << max << posr2 << posc2;
```

* Find the smallest and largest element of each row & column.

```
int a[10][10];
int r, c, min = a[0][0], max = a[0][0];
// input elements
for(i=0; i<r; ++i)   // each col
{   min = a[0][j];
    max = a[0][j];
```

```
for(j=0; j<c; ++j)
  {  if (a[i][j] < min)
     {  min = a[i][j];
     }
     if (a[i][j] > max)
     {  max = a[i][j];
     }
  }
  cout<<"for col"<<j<<" min "<<min<<" max ";
}
```

// similarly, do for row (interchange the loop)

* Swap the primary & subsidary diag:

1	2	3
4	5	6
7	8	9

⇒

3	2	1
4	5	6
9	8	7

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

⇒

5	2	3	4	1
6	9	8	7	10
11	12	13	14	15
16	19	18	17	20
25	22	23	24	21

```
for(          )
{  for(          )
   {  if(i==j){ pe = a[i][j]; }   // primary diag element
      if(i+j==n-1){ se = a[i][j]; }
   }
int t = pe;
pe = se;
se = t;
}
```

TRANSPOSE → rows become columns
columns become rows

1	2	3
4	5	6
7	8	9

→

1	4	7
2	5	8
3	6	9

// input

// transpose :→

```
for(int j=0; j<n; ++j)
    for(int i=0; i<n; ++i)
        cout<<a[i][j];
```

(same)

everything is same.
except:
interchange i loop & j loop positions

VIMP ✱ Merging of these in a few pages