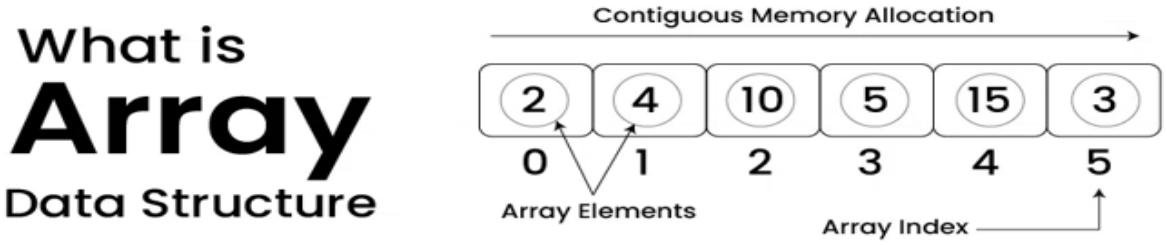
ARRAY

Array is a linear data structure where all elements are arranged sequentially. It is a collection of elements of same data type stored at contiguous memory locations.



Why is Array needed?

Ordinary variable declaration of four students for obtained marks are as follows:

int mark0=10;
int mark1=30;
int mark2=40;
int mark3=50;
Array variable declaration of four students for marks are as follows:
int mark[4]={10,30,40,50}

Variable declaration of array for marks of four students are allocated as follows:
mark[0]=10;
mark[1]=30;
mark[2]=40;
mark[3]=50;

Hence, from the above discussion array is necessity to reduce the number of ordinary variables.

Different properties of an array
There are several properties of an array. They are as follows:

1) Homogeneous data structure
Homogeneous data structure means similar types of data should be represented in one structure. The pictorial representation of several homogeneous data structures are as follows:

Integer representation:

10	20	30	40

✓ ok

Character representation:

'A'	'a'	'c'	'4'

✓ ok

Floating point representation:

10.5	20.25	30.15	40.33

✓ ok

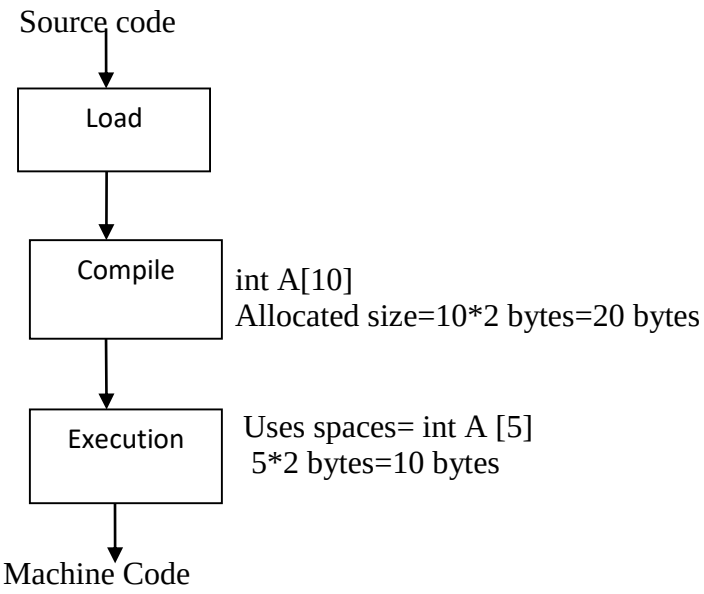
Mixed representation:

10	'A'	'3'	40.33

× Not ok

Mixed representation of data is not homogeneous. Hence, they are not supported as variable declaration in array.

2) Static memory allocation



Memory wastage= [Allocated memory size - Uses memory]
= 20-10 bytes
= 10 bytes

3) Huge memory wastage

Suppose the declaration of double type array variable A[ ] including size 100 is as follows:

double A[100]

But uses memory 40 spaces, where each space takes 8 bytes.

Therefore, total allocated memory =100*8=800 bytes
Total uses memory =40*8= 200 bytes

Wastage memory =600 bytes

4) Sequential memory allocation

We would like to store the following marks of data structure course for five students: 75, 80, 92, 84, 68.
Let the declaration of integer type array variable of size 5 is as follows:
int A[5]
The graphical representation of each mark is as follows:

values	location
75	0
80	1
92	2
84	3
68	4

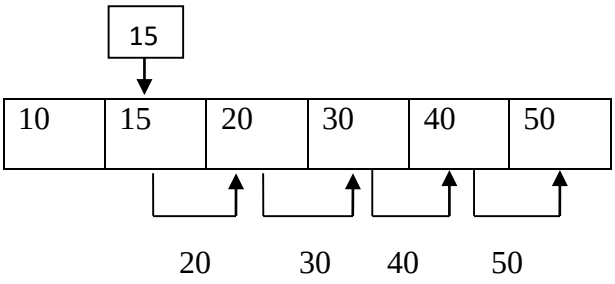
A[0] is the first location where 75 is assigned ,A[1] second one and so on. But, the question comes; how do you find the second, third and also rest of the location. The procedure is to find out the location is as follows:

loc(A[1])=loc(A[0])+2 bytes ;[size of integer data type=2 bytes]
loc(A[2])=loc(A[1)]+2 bytes
loc(A[3])=loc(A[2)]+2 bytes
loc(A[4])=loc(A[3)]+2 bytes

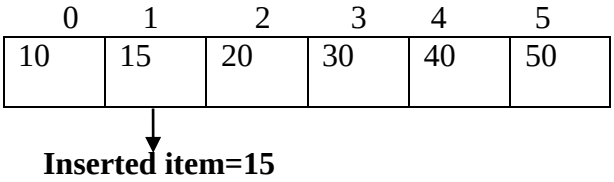
5) Difficult for insertion
Let the declaration of integer array variable of size 5 is as follows:
int A[5]
The pictorial representation of the array variable for five data is as follows:

10	20	30	40	50	
0	1	2	3	4	

If we like to insert the value 15 at location 1, the four elements (20,30,40,50) will move at right side to make room for value 15.



After Inserted item=15, the structure is below:



Total movement =4
Suppose, It takes time for one movement=C_1 sec
Hence, total time for 4 movements= 4*C_1 sec

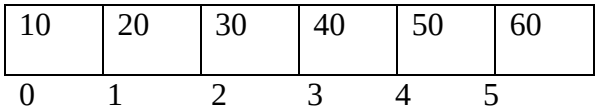
We can conclude to say that insertion data in linear array structure is difficult and time consuming.

6) Easy for binary search

Let the declaration of integer array variable of size 6 is as follows:

int A[6]

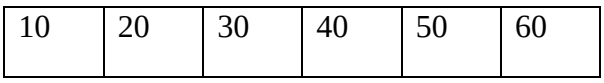
The graphical form is as follows:



Index is 0 to 5 as a sequential order. Hence the binary search is convenient.

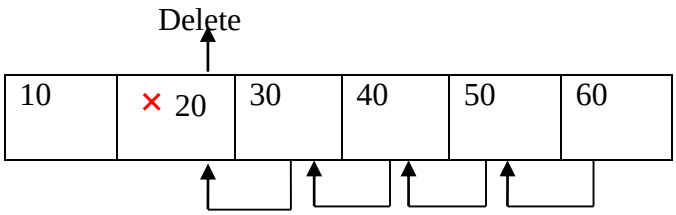
7) Difficult for deletion

Let the declaration of integer array variable of size 6 is as follows:
int A[5]
The graphical form is as follows:

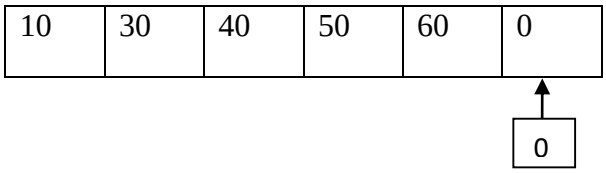


If we like to delete the value 20 at location A[1], the four elements (30,40,50,60) will move at left side since 20 is deleted.

After deleted item=20, the structure is below:



After deleted item=20, the structure is below:

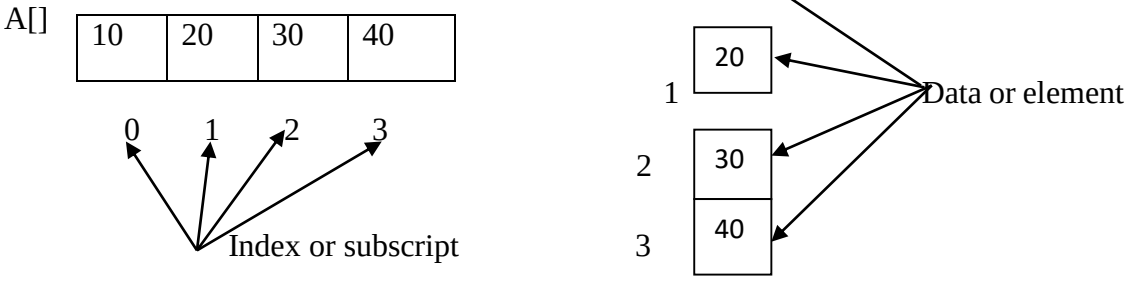


A[1]=A[2]=30
A[2]=A[3]=40
A[3]=A[4]=50
A[4]=A[5]=60
A[5]=A[6]=70
A[6]=Null=0

Different Types of Array

There are three types of array
1) 1 dimension array
2) 2 dimension array
3) 3 dimension array

1) 1 dimension array



2) 2 dimension array

Suppose three subject's marks of four students are assigned in two dimension array. The marks of three subjects are obtained by four students are as follows:

Students	Subjects		
	1st subject	2nd subject	3rd subject
1st student	80	70	68
2nd student	90	78	67
3rd student	85	87	60
4th student	82	67	86

The variable declaration of two dimension array of integer type is as follow:
int A[4][3]
For three subject's marks of four students are allocated in the following location.

Students: **Subjects**:

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

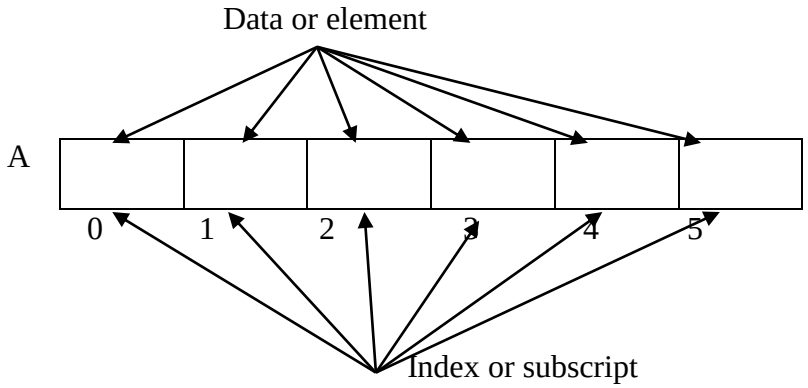
For example, marks of 1st student of 1st subject is assigned at location A[0][0], marks of 1st student of 2nd subject is assigned A[0][1] and so on. Hence, A[0][0]=80;A[0][1]=70;A[2][2]=60 and A[3][2]=86.

Variables Declaration of Array

a)Variables declaration of one dimension array for integer data type:

int A[6]

The following memory will be allocated for several integer values:



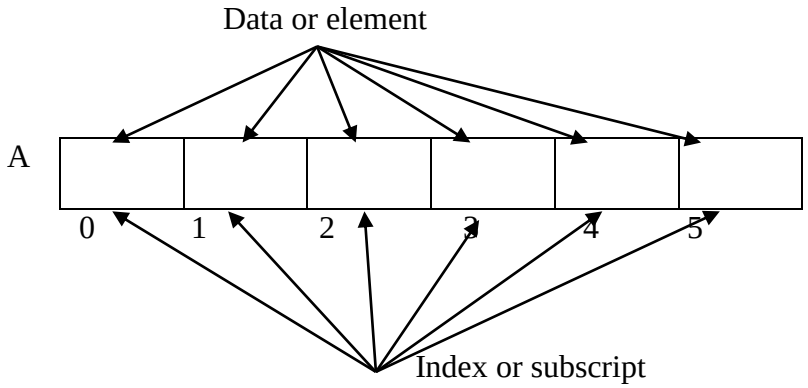
The integer values are allocated as follows:

A[0]=10;
A[1]=20;
A[2]=30;
A[3]=40;
A[4]=50;
Note: index or subscript index must be integer, but data or element might be integer, floating point, character, double etc.

b) Variables declaration of one dimension array for floating point data type:

float A[6]

The following memory will be allocated for several floating point values:

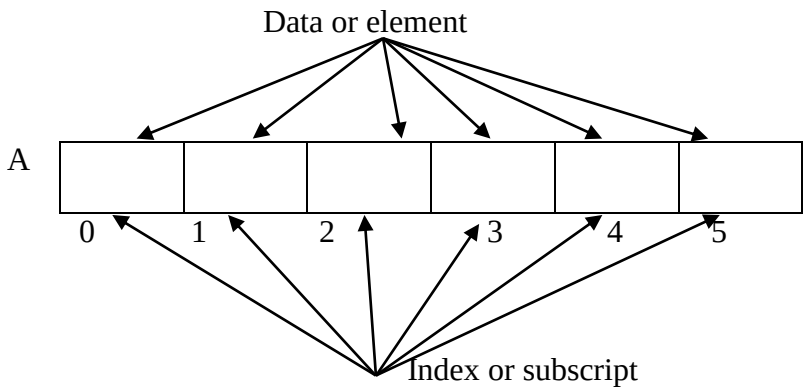


The floating point values are allocated as follows:

A[0]=10.20;
A[1]=20.15;
A[2]=30.23;
A[3]=40.25;
A[4]=50.75;

c) Variables declaration of one dimension array for character (char) data type:

char A[6]



The characters are allocated as follows:

A[0]='a';
A[1]='b';
A[2]='3';
A[3]='μ';
A[4]='A';

d) Variables declaration of two dimension array for integer (**int**) data type:

int A[4][3]

Suppose three subject's marks of four students are assigned in two dimension array. The marks of three subjects are obtained by four students are as follows:

Students	Subjects		
	1st subject	2nd subject	3rd subject
1st student	80	70	68
2nd student	90	78	67
3rd student	85	87	60
4th student	82	67	86

The variable declaration of two dimension array of integer type is as follow:

int A[4][3]

For three subject's marks of four students are allocated in the following location.

Students: **Subjects**:

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

For example, marks of 1st student of 1st subject is assigned at location A[0][0], marks of 1st student of 2nd subject is assigned A[0][1] and so on. Hence, A[0][0]=80;A[0][1]=70;A[2][2]=60 and A[3][2]=86.

e) Variables declaration of two dimension array for floating point (**float**) data type:

float A[4][3]

Suppose three month's salaries of four employees are assigned in two dimension array. The salaries of three months are obtained by four employees are as follows:

Employees	Months		
	January	February	March
1st employees	28800	30000	29700

2nd employees	34000	32000	33000
3rd employees	34150	30700	27000
4th employees	28700	32600	32000

The variable declaration of two dimension array of floating point type is as follow:

float A[4][3]

For three subject's marks of four students are allocated in the following location.

Employees**:** **Months**:

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

f) Variables declaration of two dimension array for character (char) data type:

char A[4]3[]

Suppose three subject's grades of four students are assigned in two dimension array. The grades of three subjects are obtained by four students are as follows:

Students	Subjects		
	1st subject	2nd subject	3rd subject
1st student	B	C^-	B^+
2nd student	A^+	C^+	A
3rd student	A^-	A^-	B
4th student	B^+	C	A^-

The variable declaration of two dimension array of character (char) type is as follow:
char grade[4][3]
For three subject's grades of four students are allocated in the following location.

Students**:** Subjects

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

Some Simple Programs using Array

Some Simple programs of one dimension array:

a) Find average of n numbers stored in array a[ ].

```
#include <stdio.h>
void main()
{
   int n, i;
   float num[100], sum=0.0, average;
   printf("Enter the numbers of data(n): ");
   scanf("%d",&n);
  for(i=0; i<n; ++i)
  {
    printf("%d. Enter number: ",i+1);
    scanf("%f",&num[i]);
    sum+=num[i];
  }
```

```
   average=sum/n;
   printf("Average = %.2f",average);
   //return 0;
}
```

b) To insert an element in array **num[ ]** where data are in ascending order.

```
#include <stdio.h>
#include<stdlib.h>
void main(){
    int n, i,loc,k;
    float num[100], item;
    printf("Enter the numbers of data(n): ");
    scanf("%d",&n);
   for(i=0; i<n; ++i)
   {
      printf("%d. Enter number: ",i+1);
      scanf("%f",&num[i]);
   }
   for(i=0; i<n-1; ++i)
            {
            if(num[i]>num[i+1])
              {
 printf("Error!! You should mind your documentation!!\n");
              exit(0);
   }
}
printf("The input data elements before inserted the item")
   for(i=0; i<n; ++i)
```

```
  {
  printf("%.2f\t",num[i]);
   }
printf("\n");
printf("Enter the value of item:");
scanf("%f",&item);
loc=n;
for(i=0;i<n;++i)
{
if(num[i]>item)
{
            loc=i;
            break;
}
}
for(k=n-1;k>=loc;k--)
{
num[k+1]=num[k];
}
num[loc]=item;
printf("Inserted Item: %.2f\n",num[loc]);
n=n+1;
printf(“The output data elements after inserted the item”);
for(i=0;i<n;++i)
{
printf("%.2f\t",num[i]);
}
```

```
printf("\n Number of element= %d\n",n);

}
```

Output:

```
Enter the number of data(n):3

1.Enter number:13

2.Enter number:15

3.Enter number:18

The input data elements before inserted the item:13.00 15.00 18.00

Enter the value of item:17

Inserted item:17.00

The output data elements after inserted the item:13.00 15.00 17.00 18.00

Number of element=4
```

c) To delete an element from an array **num[]**

```
#include <stdio.h>

void main()

{

   int n, i,loc,k;

   float num[100], item , null=0;

   printf("Enter the numbers of data(n): ");

   scanf("%d",&n);

  for(i=0; i<n; ++i)

  {

    printf("%d. Enter number: ",i+1);

    scanf("%f",&num[i]);

  }
```

```
  printf("The input data elements before deleted the item:");
   for(i=0; i<n; ++i)
   {
   printf("%.2f\t",num[i]);
    }
   printf("\n");
printf("Enter the value of item:");
scanf("%f",&item);
for(i=0;i<n;++i){
            if(num[i]==item)
            {
                        loc=i;
                        break;
            }
}
for(k=loc;k<=n-2;++k){
num[k]=num[k+1];
}
num[n-1]=null;
n=n-1;
printf("\nThe output data elements after deleted the item:");
for(i=0;i<n;i++)
{
            printf("%.2f\t",num[i]);
}
printf("\n Finally Number of element= %d\n",n);
}
```

d) To merge two array A[ ] and B[ ].

```
#include<stdio.h>
#include<stdlib.h>
void main()
{           int i,j,n,m,p,k;
            float a[100],b[100],c[100];
            printf("Enter the number (n):");
            scanf("%d",&n);
            for(i=0;i<n;i++)
            {
                        printf("%d. Enter the number:",i+1);
                        scanf("%f",&a[i]);
            }
            for(i=0; i<n-1; ++i)
            {
            if(a[i]>a[i+1])
              {
 printf("Error!! You should mind your documentation\n");
              exit(0);
 }
            }
            printf("The values of array A:");
                        for(i=0;i<n;i++)
            {
                        printf("%.2f\t",a[i]);
            }
                        printf("\n");
```

```
                printf("Enter the number (m):");
                scanf("%d",&m);
                for(j=0;j<m;j++)
        {
                printf("%d. Enter the number:",j+1);
                scanf("%f",&b[j]);
        }
        for(j=0; j<m-1; ++j)
        {
        if(b[j]>b[j+1])
          {
printf("Error!! You should mind your documentation\n");
          exit(0);
      } }
        printf("The values of array b:");
                for(j=0;j<m;j++)
        {
                printf("%.2f\t",b[j]);
        }
        printf("\n");
                i=0;
                j=0;
                k=0;
                while((i<n) &&(j<m))
                {
                        if(a[i]>b[j])
                        {
```

```
                    c[k]=b[j];
                    j=j+1;
                    k=k+1;
                }
                    else
                    {
                    c[k]=a[i];
                    i=i+1;
                    k=k+1;+
                    }
            }
            if(i>=n)
                    for(p=j;p<m;p++)
                    {
                    c[k]=b[p];
                    k=k+1;
                    }
                    else
                    for(p=i;p<n;p++)
                    {
                       c[k]=a[p];
                       k=k+1;
            }
printf("\n After merge the new array:");
for(k=0;k<(n+m);k++)
{
printf("%.2f\t",c[k]);
```

```
}
printf("\n");
}
```

Simple program of two dimension array:

a) C program for Matrix addition:

```
#include<stdio.h>
void main()
{
                    int i,n,j,k,p;
                    int b[5][5]={0};
                    int c[5][5]={0};
                    int a[5][5]={0};
  printf("Enter the value of n:");
  scanf("%d",&n);
            for(i=0;i<n;i++)
    {
                    for(j=0;j<n;j++)
                {
  printf("Enter the %d input of A matrix:",j+1);
  scanf("%d",&a[i][j]);
                    }
          }
  printf("The A matrix:\n");
                    for(i=0;i<n;i++)
                    {
```

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

        for(i=0;i<n;i++)
        {
                for(j=0;j<n;j++)
                {
  printf("Enter the %d input of A matrix:",j+1);
   scanf("%d",&b[i][j]);
                }
        }
printf("The B matrix:\n");
        for(i=0;i<n;i++)
        {
                for(j=0;j<n;j++)
                {
                printf("% d",b[i][j]);
                }
                printf("\n");
        }
        for(i=0;i<n;i++)
        {
```

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


                    }
                }
        printf("The summation  matrix C:\n");
                for(k=0;k<n;k++)
                {
                    for(p=0;p<n;p++)
                    {

                 printf("%5d",c[k][p]);
                    }
                   printf("\n");
                }
}
```

Output:

```
Enter the value of n:2
Enter the 1 input of A matrix:1
Enter the 2 input of A matrix:2
Enter the 1 input of A matrix:3
Enter the 2 input of A matrix:4
The A matrix:
 1    2
 3    4
Enter the 1 input of B matrix:4
Enter the 2 input of B matrix:3
Enter the 1 input of B matrix:2
```

```
Enter the 2 input of B matrix:1
The B matrix:
 4    3
 2    1
The summation matrix C:
 5    5
 5    5
```

b) C program for Matrix multiplication:

```
#include<stdio.h>

void main()

{

        int i,n,j,k,p;

        int b[5][5]={0};

        int c[5][5]={0};

        int a[5][5]={0};

   printf("Enter the value of n:");

        scanf("%d",&n);

        for(i=0;i<n;i++)

        {

                for(j=0;j<n;j++)

                {

    printf("Enter the %d input of A matrix:",j+1);

        scanf("%d",&a[i][j]);

                }

        }

   printf("The A matrix:\n");

                for(i=0;i<n;i++)

                {
```

```
                for(j=0;j<n;j++)
                {
                        printf("% d",a[i][j]);
                }
                printf("\n");
        }
        for(i=0;i<n;i++)
        {
                for(j=0;j<n;j++)
                {
printf("Enter the %d input of B matrix:",j+1);
                 scanf("%d",&b[i][j]);
                 }
        }
printf("The B matrix:\n");
        for(i=0;i<n;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf("% d",b[i][j]);
                }
                printf("\n");
        }

        for(i=0;i<n;i++)
        {
```

```
                for(j=0;j<n;j++)
                {
                        c[i][j]=0;
                        for(k=0;k<n;k++)
                        {
                         c[i][j]=c[i][j]+a[i][k]*b[k][j];
                        }
                }
        }
        printf("The multiplication  matrix C:\n");
        for(k=0;k<n;k++)
        {
                for(p=0;p<n;p++)
                {
               printf("%5d",c[k][p]);
                }
                printf("\n");
        }
}
```

Output:

```
Enter the value of n:2
Enter the 1 input of A matrix:1
Enter the 2 input of A matrix:2
Enter the 1 input of A matrix:3
Enter the 2 input of A matrix:4
The A matrix:
 1   2
 3   4
Enter the 1 input of B matrix:5
Enter the 2 input of B matrix:4
Enter the 1 input of B matrix:4
Enter the 2 input of B matrix:5
```

The B matrix:
5 4
4 5
The multiplication matrix C:
13 14
31 32

d) C program for magic square:

```
#include<stdio.h>

void main()

{

        int x,y,i,n,j,k;

        int a[10][10]={0};

        n=3;

        y=n/2;

        x=0;

        a[x][y]=1;

for(i=2;i<=n*n;i++)

{

        if(((x-1)<0)&&((y+1)<=(n-1)))

        {

                a[n-1][y+1]=i;

           x=n-1;

           y=y+1;

        }

   else if (((x-1)>=0) && ((y+1)>(n-1)))

        {

                a[x-1][0]=i;
```

```
                x=x-1;
                y=0;
        }
        else if ((((x-1)>=0) && ((y+1)<=n-1)) && (a[x-1][y+1]>0))
        {
                a[x+1][y]=i;
               x=x+1;
        }
        else if ((((x-1)>=0) && ((y+1)<=(n-1))) && (a[x-1][y+1]==0))
        {
                a[x-1][y+1]=i;
                x=x-1;
                y=y+1;
        }
        else if (((x-1)<0) && ((y+1)>(n-1)))
        {
                a[x+1][y]=i;
           x=x+1;
        }
}
for(k=0;k<n; k++)
{
        for(j=0;j<n; j++)
        {
        printf("    [%d]",  a[k][j]);
```

```
        }

        printf("\n");

}

return;

}
```

Output:

The Magic Square:

[8] [1] [6]
[3] [5] [7]
[4] [9] [2]

Memory Allocation of Array

What is memory allocation?

Objectives of memory allocation:

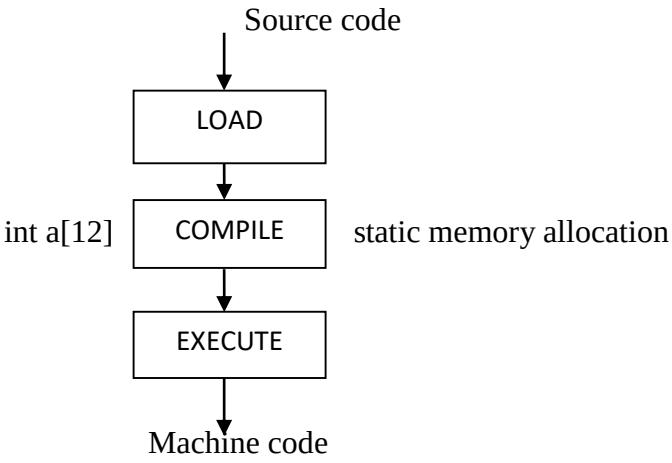
- ✓ To understand the differences between static and dynamic memory allocation
- ✓ To manage dynamic memory
- ✓ To analyze memory management related bugs.

Types of memory allocation:

- → Static memory allocation
- → Dynamic memory allocation

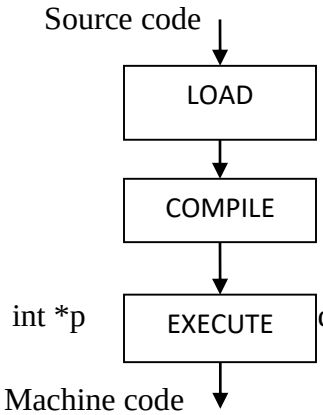
Difference between static and dynamic memory allocation:

- → Static allocation allocates memory at compile time.



→ Dynamic allocation allocates memory at execution time.

.



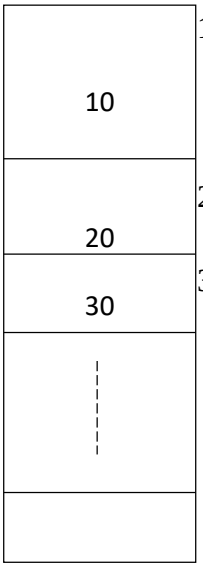
Array basically allows static memory allocation.

Allocation for 1 D array:

The variable declaration of 1D array is as follows:

int A[n]

A



If integer data width, w=2 bytes

loc(A[3])=loc(A[1])+2 bytes+2bytes

= loc(A[1]+4

=loc(A[1])+2*2

=loc(A[1])+2*(3-1)

↓

Integer data width (w)

In general, replace 3 by i we have,

loc(A[i])=loc(A[1]) +w*(i-1)

= loc(A[base]) +w*(i-base) ;[1=base from array A]

Allocation formula for one dimensional array is as follows:

loc(A[i])=loc(A[base])+w*(i-base)

For example: The declaration of double type one dimensional array variable is the following:
double A[100]
if loc(A[5])=$5CDFE_{16}$
Find loc(A[80])
Solution:
For double, w=8 bytes
base=5
loc(A[5])=$5CDFE_{16}$
i=80
We know that, loc(A[i])=loc(A[base])+w*(i-base)

loc(A[80])=loc(A[5])+ 8 *(80-5)

$=5CDFE_{16}$ +8*75

$=5CDFE_{16}$ +600_{10}

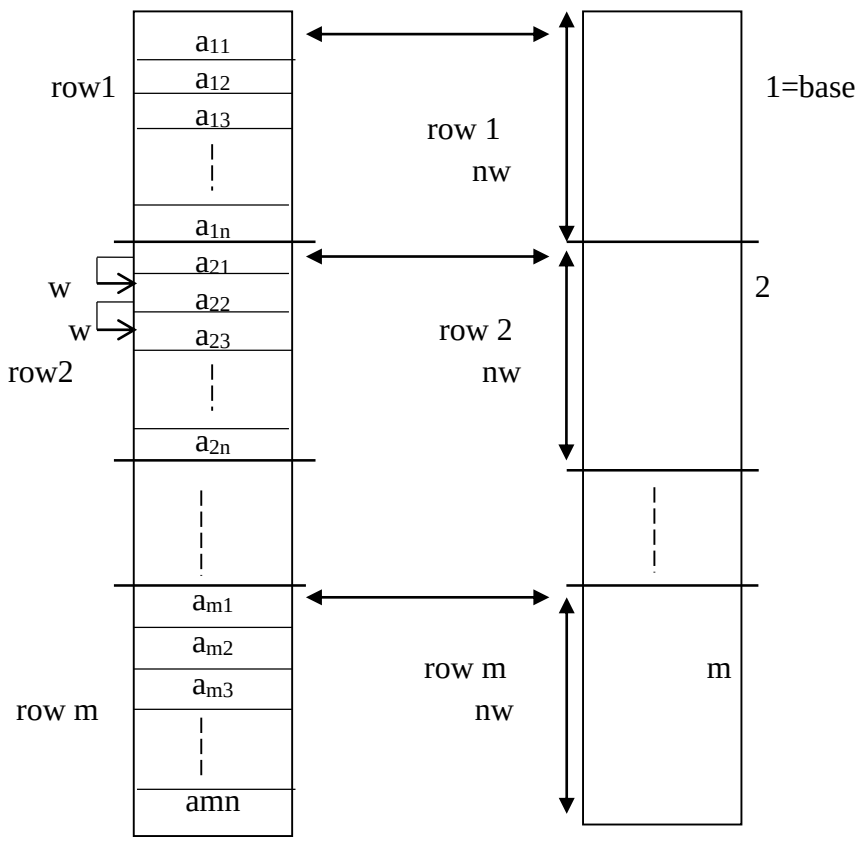
$=5CDFE_{16}$ +258_{16}

$=5D056_{16}$

Memory allocation for two dimensional array (row wise):

Let following the matrix:

a_{11} a_{12} a_{13} ……a_{1n}
a_{21} a_{22} a_{23}……..…..a_{2n}
a_{21} a_{32} a_{33} ……..…..a_{3n}
⋮
a_{m1} a_{m2} a_{m3}……a_{mn}



We know that for one dimension array,
loc(A[i])=loc(A[base])+w*(i-base)
For row 2 of array A we have =>
loc(A[2])=loc(A[1])+nw*(2-1)

For row 2 and column 3 of array A we can have =>
$loc(a_{23})=loc(a_{21})+w+w$
$loc(a_{23})=loc(a_{21})+2w$
$=loc(a_{21})+(3-1)w$
$=loc(A[2])+(3-1)w$
$=loc(A[1])+nw*(2-1)+(3-1)w$
$=loc(a_{11})+w[(2-1)n+(3-1)]$
In general when 2=i row and 3=j column,
$loc(a_{ij})=loc(a_{11})+w[(i-1)n+(j-1)]$
$=loc(a_{\text{r base, c base}})+w[(i-\text{r base})n+(j-\text{c base})]$

Example:
double A[70][80];
if $loc(A[0][0])=5CDFE_{16}$
Find loc(A[50][60])
Solution:
For double, w=8 bytes
Given that, row base=0, column base=0
n=80; [no. of column]
i=50
j=60
$loc(A[0][0])=5CDFE_{16}$
From allocation formula of two dimensional array =>
loc(A[i][j])=loc(A[r base][c base])+w[(i-r base)n+(j- c base)]

loc(A[50][60]) =loc(A[0][0])+8[(50-0)*80+(60-0)]
$=5CDFE_{16}+8[4000+60]$
$=5CDFE_{16}+32480_{10}$
$=5CDFE_{16}+7EE0_{16}$
$=64CDE_{16}$

Problem:
Find the memory location of A[70][80] if loc(A[15][20])=x+1400, where x=last four digits of your student ID. Assume column-wise memory is allocated in the double type array A[90][100], where each double data is 8 bytes.