Practical-6

To study about types of arrays, declaration of arrays and their initialization with their applications.

Practical-6.1

Write program which scans five values and store it consecutive position of an array and display it like this:

a[0]=value1

a[1]=value2

a[2]=value3

a[3]=value4

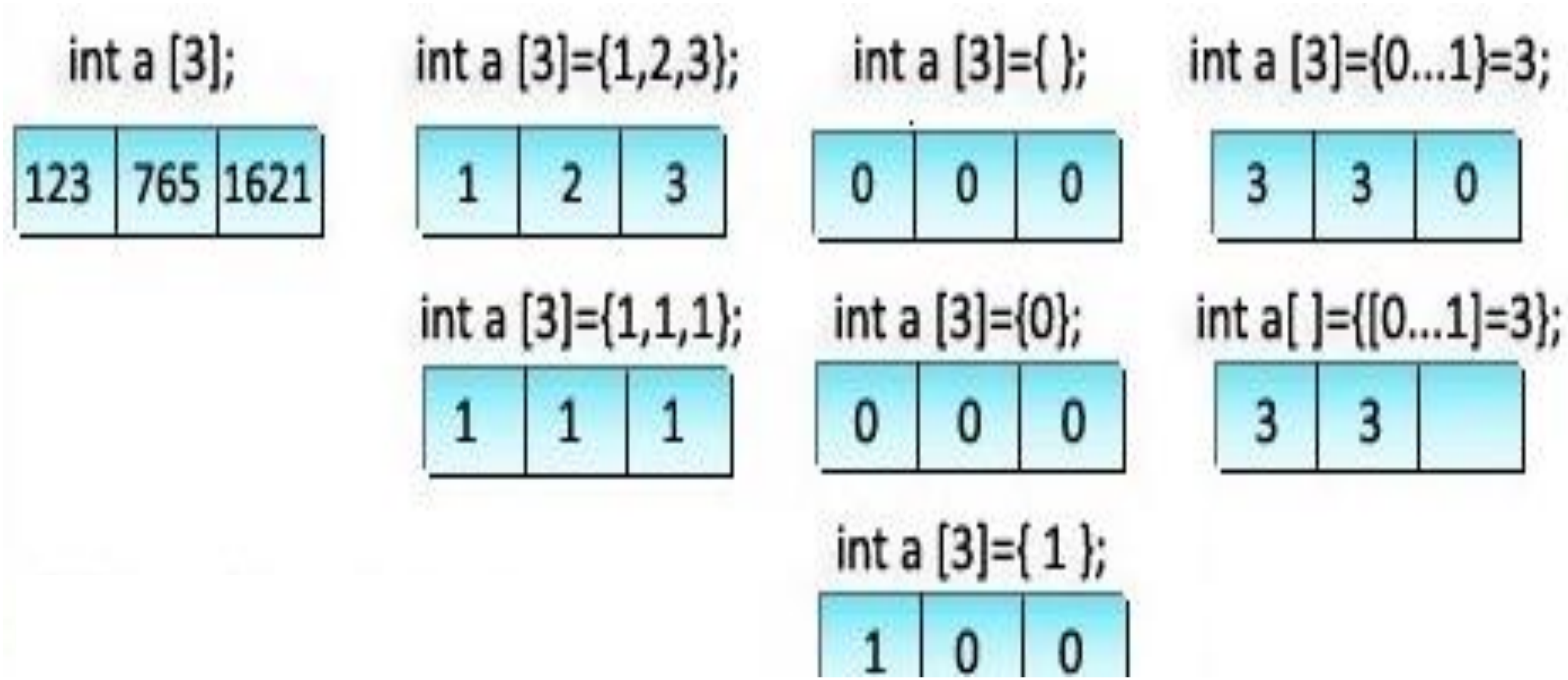
a[4]=value5

1000	45
1004	67
1008	
1012	

Array

- An **array** is a collection of data items, all of the same type, accessed using a common name.
- **Array** size must be a constant value
- **Declaration**
- Datatype ArrayName Size
- E.g
- Int Number[10];
- size and type of an array cannot be changed once it is declared.

Array Declaration



int a [3];

123	765	1621

int a [3]={1,2,3};

1	2	3

int a [3]={ };

0	0	0

int a [3]={0...1}=3;

3	3	0

int a [3]={1,1,1};

1	1	1

int a [3]={0};

0	0	0

int a[ ]={[0...1]=3};

3	3	

int a [3]={ 1 };

1	0	0

Access Array Element

- You can access elements of an array by indices.
- Arrays have 0 as the first index
- If the size of an array is n , to access the last element, the n-1 index is used.
- E.g int Number[5];
- 1st element is Number[0] and last element is Number[4]

Change value of Array Element

- Number[0]=1;
- Number[4]=43;
- Number[0]=21;

Input and Output Array Elements

- Scanf(“%d”,&Number[2]);
- Enter value for element Number[2] only
- Input value for all element

```
For(i=0;i<10;i++)
    {
        scanf(“%d”,&Number[i]);
    }
```

Practical-6.1

1. Start
2. Declare array
3. Read Array Values
4. Print Array Values
5. End

Practical-6.2

- Write a program to display sum and average of any 10 Numbers using 1-D Array.
  - Using int array
  - Using float array

Practical-6.2

1. Start
2. Declare array
3. Read array value
4. Calculate sum Average
5. Print sum and Average
6. End

Practical-6.3

- Write a program to calculate
  - sum of all the elements located on odd indexes and
  - multiplication of all elements on even indexes

Practical-6.3

- Start
- Enter size of array
- Declare array
- Read array Value
- For each i in Array
    - i is odd calculate sum
    - I is even calculate multiplication
- Print sum and multiplication value
- End

Practical-6.4

- Write a Program to Find largest odd number from given 1-D Array.

1. Start
2. Read array element
3. For each array element
    - Check element value is odd
        - Check value is max or not
4. Print max value
5. End

Practical-6.5

- Write a program to sort the array (arranging elements in ascending order)
  - int array
  - character array

Character Array

- Char name[20]=“XXYYZZ”;

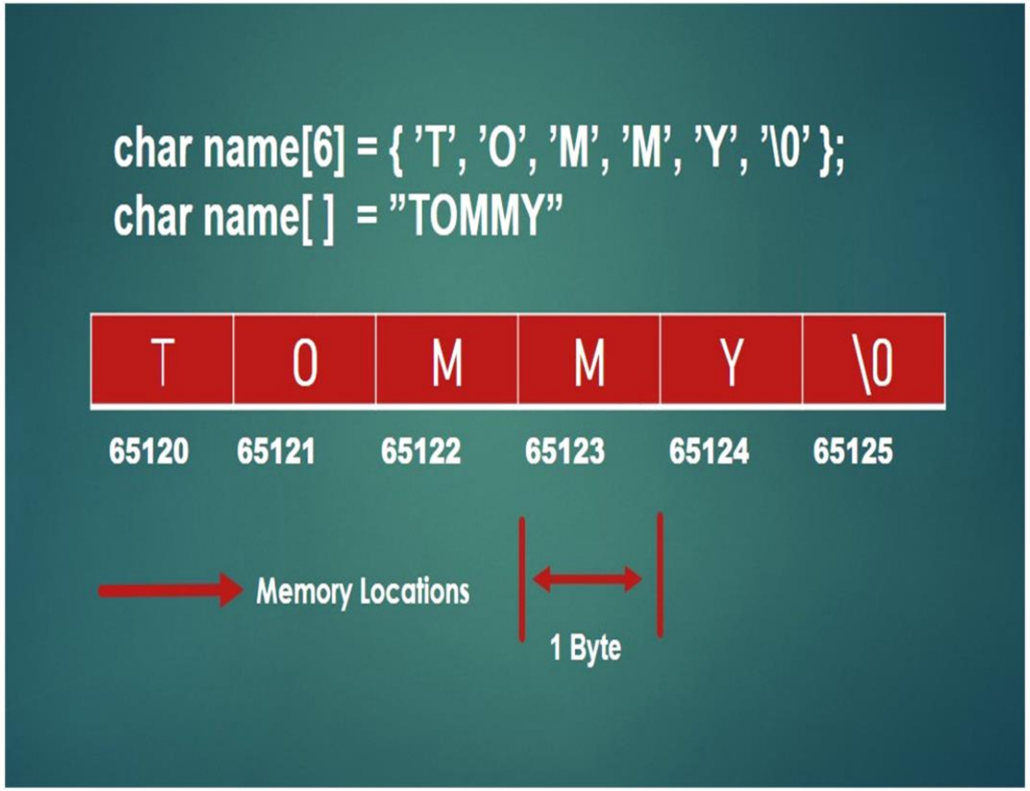
//occupied 7-byte memory string length=6

- Char name[]=“XXX YYY”;

//required 10 byte of memory,string length=7

- Char ch[5]={‘p’ , ’r’, ‘t’,’s’,’\0’};

- Char ch[10]=“Ganpat university”;



```
char name[6] = { ’T’, ’O’, ’M’, ’M’, ’Y’, ’\0’ };
char name[ ]  = ”TOMMY”
```

Practical-6.5

- Start
- Declare array
- Read array value
- Compare each element(i) with nextj=(i+1) element
    - Sort in ascending order
- Print all element of sorted Array
- End

Practical-6.6

- Perform following operation on two dimensional dynamically initialized matrices of size 3x3
  1. C = A + B
  2. C = A x B

2 Dimensional Array

- Declaration
  - Arrayname [R_size][C_size];
- Access Array Element
  - **int** x = a[i][j];
- Initialize Array
  - **int** arr[2][2] = {0,1,2,3};

Practical-6.6

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}$$

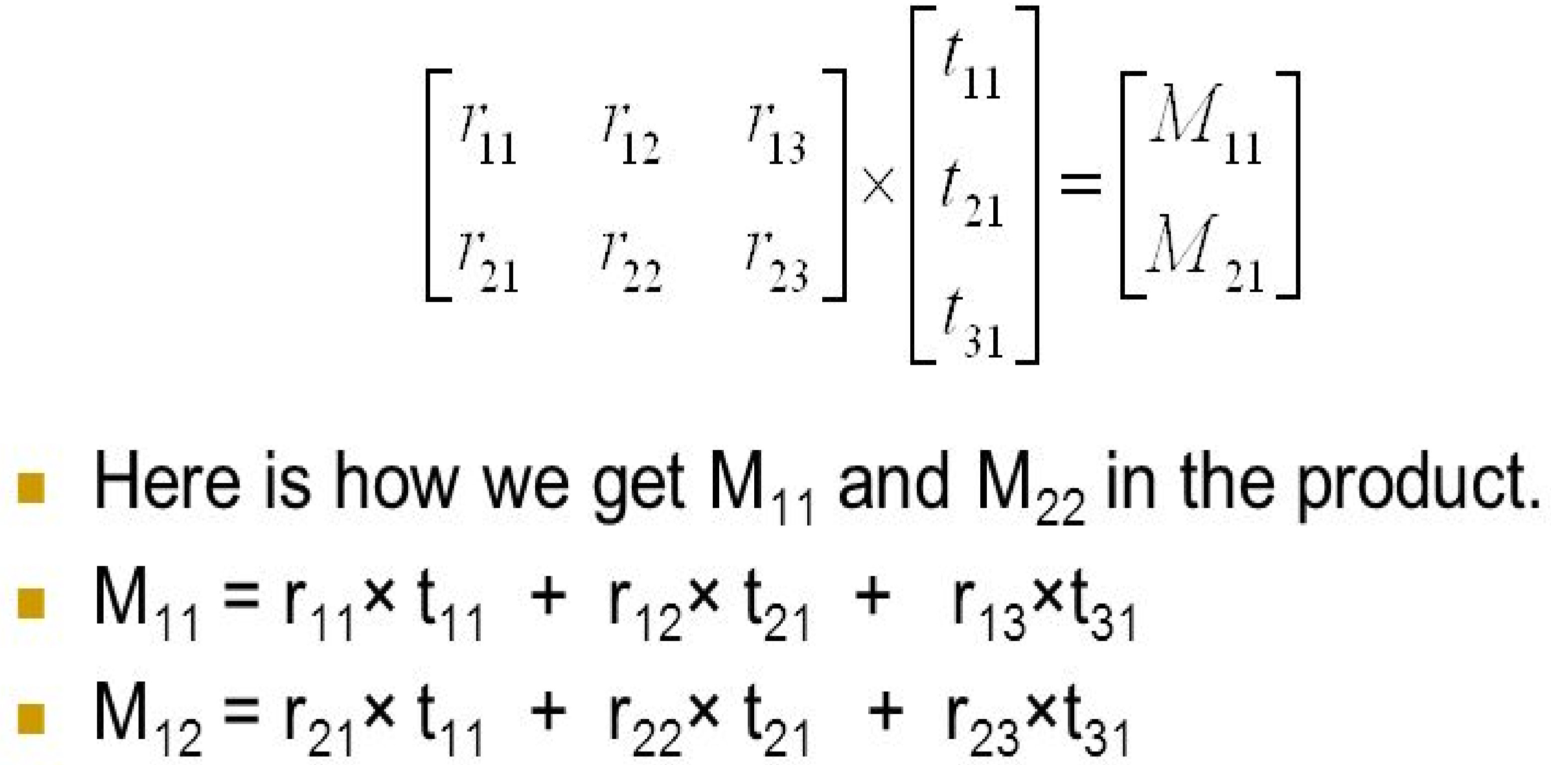
Matrix Multiplication

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} \text{ x } \begin{bmatrix}10 & 11\\ 20 & 21\\ 30 & 31\end{bmatrix}$$

$$= \begin{bmatrix}1\text{x}10 + 2\text{x}20 + 3\text{x}30 & 1\text{x}11 + 2\text{x}21 + 3\text{x}31\\ 4\text{x}10 + 5\text{x}20 + 6\text{x}30 & 4\text{x}11 + 5\text{x}21 + 6\text{x}31\end{bmatrix}$$

$$= \begin{bmatrix}10+40+90 & 11+42+93\\ 40+100+180 & 44+105+186\end{bmatrix} = \begin{bmatrix}140 & 146\\ 320 & 335\end{bmatrix}$$

Matrix Multiplication



$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \end{bmatrix} \times \begin{bmatrix} t_{11} \\ t_{21} \\ t_{31} \end{bmatrix} = \begin{bmatrix} M_{11} \\ M_{21} \end{bmatrix}$$

- Here is how we get M_{11} and M_{22} in the product.
- $M_{11} = r_{11}\times t_{11} + r_{12}\times t_{21} + r_{13}\times t_{31}$
- $M_{12} = r_{21}\times t_{11} + r_{22}\times t_{21} + r_{23}\times t_{31}$

Matrix Multiplication

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$\begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} & a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} \end{bmatrix}$$

- c[i][j] = c[i][j] + a[i][k] * b[k][j];

Practical-6.7

- Write a program to find transpose of 2-D matrix.
- A[0][0]=B[0][0],A[0][1]=B[1][0],A[1][0]=B[0][1],A[1][1]=B[1][1]
- B[J][I]=A[I][J]

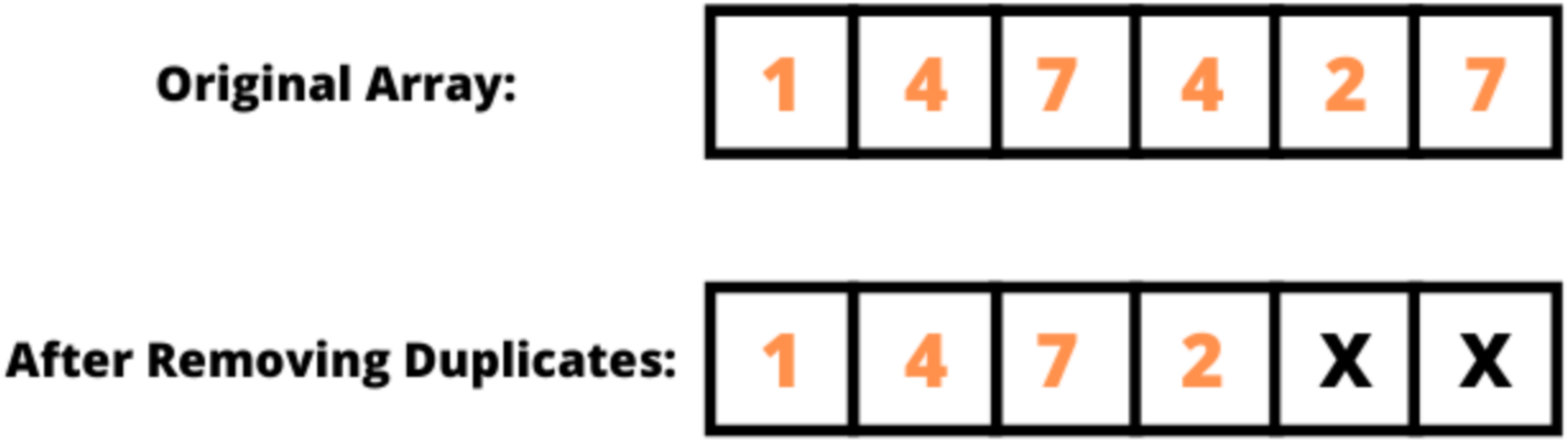
A	0	1
0	X	Y
1	Z	W

→

B	0	1
0	X	Z
1	Y	W

Practical - 6.8

- Write a program to find and remove duplicates elements from an array.



Original Array:	1	4	7	4	2	7
After Removing Duplicates:	1	4	7	2	X	X

Practical - 6.9

- Write a program which finds the Minimum element from one dimensional int array.
- **OUTPUT:**
- Minimum element is present at location __ with value:__