UNIT-VII
Arrays

Definition:

Array is a fixed size sequenced collection of elements of the same data type

Example:

1. List of employees in an organization
2. Test scores of a class of students

Types of Arrays

1. One-Dimensional Array
2. Two-Dimensional Array
3. MultiDimensional Array

One-Dimensional Array

Definition:

A list of items can be given one variable name using **only one subscript** and such a variable is called a **single-subscripted variable or a one-dimensional array**

In a single subscripted variable xi can be expressed as

x[1],x[2],x[3],.....x[n]

Always array subscript begins with x[0] and ends with x[n-1]

Example:

If we want to represent a set of 5 numbers,say(10,20,30,50,60) by an array variable number

Declaration is as follows

```
int number[5];
```

Declaration of one-dimensional arrays

Array must be declared before they are used so that compiler can **allocate space for them in memory**

Syntax:

datatype variable_name[size];

where
datatype :specifies type of element like int,float

size: indicates the maximum number of elements that can be stored inside the array

Example:

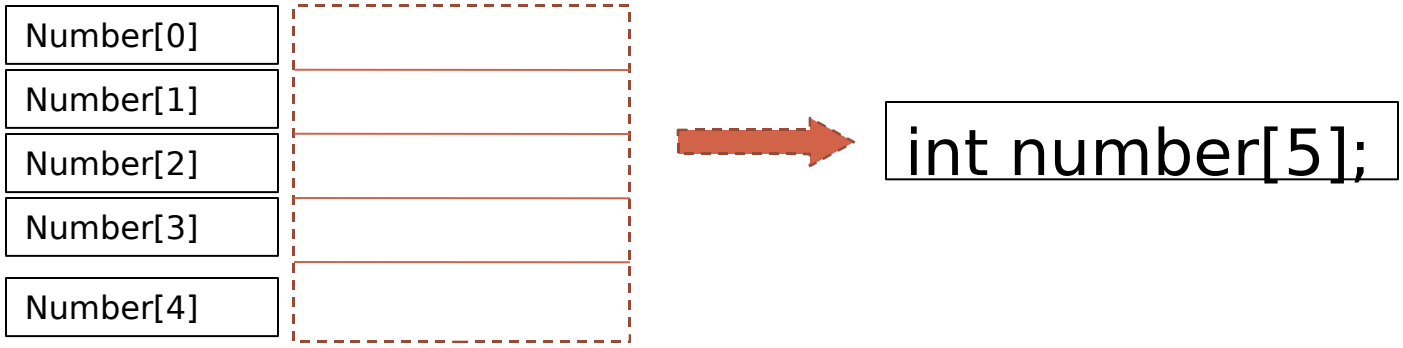
1. **float height[50];**

declares the height to be an array containing 50 real elements

2. **int group[10];**

declares group as an array to contain a maximum 10 integer constants

And the computer reserves 5 storage locations as shown below



Two dimensional arrays:

There are certain situations where a table of values will have to be stored

C allows us to define such table using two dimensional arrays.

Two dimensional arrays are declared as follows:

Type array_name [row_size][column_size];

In c language the array sizes are separated by its own set of brackets.

1. Two dimensional arrays are stored in memory as shown in the table below.
2. Each dimension of the array is indexed from zero to its maximum size minus one;
3. the first index selects the row and the second index selects the column within that row.

	Column0	Column1	Column2
	[0][0]	[0][1]	[0][2]
Row 0	210	340	560
	[1][0]	[1][1]	[1][2]
Row 1	380	290	321
	[2][0]	[2][1]	[2][2]
Row2	490	235	240
Row3	[3][0]	[3][1]	[3][2]
	240	350	480

Initialization of two-dimensional arrays:

The general form of initialization of arrays is:

static type array-name[row][col]={ list of values};

The values in the list are separated by commas.
For example, the statement below shows

```
int num[3][3]={2,2,2,2,2,2,2,2,2};
2  2  2
2  2  2
2  2  2
```

1. Will declare the variable num as an array of size 3*3 and will assign 2 to each element.
2. If the number of values is less than the number of elements, then only that many elements will be initialized.
3. The remaining elements will be set to zero automatically.

For example:

static float num1[3][2]={0.1,2.3,4};

0 1

2 3

4 0

1. Will initialize the first five elements to 0.1,2.3 ,4 and the remaining one elements to zero.
2. The word static used before type declaration declares the variable as a static variable.
3. In some cases the size may be omitted. In such cases, the compiler allocates enough space for all initialized elements.

   For example, the statement

1. Program to read and write two dimensional arrays.

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

main()
{
int a[10][10];
 int i, j , row, col;

          printf(“\n Input row and column  of a matrix:”);
          scanf(“%d %d”,  &row,&col);
          for(i=0; i<row;i++)
          for(j=0;j<col;j++)
          scanf(“%d”, &a[i][j]);
          for(i=0;i<row;i++)
{
                    for(j=0;j<col;j++)
                    printf(“%5d”, a[i][j]);
printf(“\n”);
}
}
```

2. Program to find the sum of diagonal elements of two-dimensional arrays.

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

main()
{
int a[10][10];
          int i, j , row, col, sum=0;
             printf("\n Input row and column  of a matrix:");
             scanf("%d %d",  &row,&col);
             for(i=0; i<row;i++)
             for(j=0;j<col;j++)
             scanf("%d", &a[i][j]);
          printf("The given array elements are \n");
             for(i=0;i<row;i++)
{
                          for(j=0;j<col;j++)
                          printf("%5d", a[i][j]);
printf("\n");
}
          for(i=0; i<row; i++)
             for(j=0;j<col ; j++)
          if (i = =j)
          sum = sum + a[i][j];
          printf("The sum of the diagonal elements are %d ",sum);
getch();
}
```

1. Write a C program to multiply into matrices A & B.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
main()
{
int m, n, p, q, i, j, k, a[10][10], b[10][10], c[10][10];
printf(“Enter the order of matrix A \n”);
scanf(“%d %d”, &m, &n) ;
printf(“Enter the order of matrix B \n”);
scanf(“%d %d”, &p, &q);
if(n!=p)
{
printf(“Multiplication  is not possible”);
exit();
}
else
printf(“Enter elements of matrix A \n”);
for(i=0;i<m;i++)
for(j=0;j<n;j++)
scanf(“%d”, &a[i][j]);
```

```
printf(“Enter elements of matrix B”);
for(i=0;i<p;i++)
for(j=0;j<q;j++)
scanf(“%d”, &b[i][j]);
for(i=0;i<m;i++)
{
for(j=0;j<q;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 resultant matrix C is \n”);
for(i=0;i<m;i++)
{
for(j=0;j<q;j++)
{
printf(“%d”,c[i][j]);
}
printf(“\n”);
}
   }
}
```