Array

An array can be defined as no of memory location each of which can store the same data type and which can be referenced to the same variable name.

An array is a collective name given to the group of similar quantity. A specific element in an array is accessed by an index. These similar quantity could be percentage marks of students, no. of chairs in home or salaries of 500 employees or age of 10 students. Thus an array is a collection of similar elements. These similar elements could be all integers or all characters or usually all floats.

Usually the array of characters is called a string, whereas an array of integers or floats is called simply an array. All elements of any given array must be of the same type i.e. we can't have an array of 5 integers and 5 floats.

We can also use pointer in array for different memory location.

Declaration of an array

Type variable_name[size]

char name[200]

float width[50]

double height[70]

int marks[100]

marks[4] =50

assigns element number 5 in marks

In C, all arrays have 0 as the index of their first element. Therefore, when you write

char name[10];

We are declaring a character array that has 10 elements, **p[0]** through **p[9]**. For example, the following program loads an integer array with the numbers 0 through 99:

int min[5] = {10,20,30,40,50}

To point all the elements of the array

```
for( int i<0; i<5; i++)
{
        printf("%d", min[i]);
}
```

Initializing Array

int min[5] = {1,2,3,4,5};

```
int min[] = {1,2,3,4,5};


int age[6];
age[0] = 20;
age[1] = 22;
age[2] = 24;
age[3] = 20;
age[4] = 20;
age[5] = 26;
```

All the elements in an array would store in contiguous memory location

Element	age[0]	age[1]	age[2]	age[3]	age[4]	age[5]
Address	65822	65824	65826	65828	65830	65832

Q) Average marks of 20 students using array

```
#include<stdio.h>
int main()
{
	int marks[20],avg;
	int sum=0;
	printf("Enter the marks of 20 students\n");
	int i;
	for(i=0;i<20;i++)
	{
		printf("\nEnter marks");
		scanf("%d",&marks[i]);
	}
	for(i=0;i<20;i++)
	{
		sum = sum+marks[i];
	}
```

```
	avg=sum/20;
	printf("\nAverage = %d",avg);
	return 0;
}
```

There is no bound checking in C. we could overwrite either end of an array and write into some other variable's data or even into the program's code. As the programmer, it is our job to provide bounds checking where needed. For example, this code will compile without error, but it is incorrect because the **for** loop will cause the array **marks** to be overrun.

```
main()
{
	int marks[50],i
	for(i=0;i<100;i++)
	marks[i]=i
}
```

Passing Single-Dimension Arrays to Functions

In C, we cannot pass an entire array as an argument to a function, however, we can pass the array elements by the calling function to called function. For e.g.

In Call by Vaule

```
#include<stdio.h>
main()
{
	int i,marks[]={10,30,20,50,25};

	for(i=0;i<5;i++)
	{
		display(marks[i]);
	}
	return 0;
}

display(int m)
{
	printf("%d\n",m);
}
```

In call by reference

```
main()
{
	int i,marks[]={10,30,20,50,25};

	for(i=0;i<6;i++)
	display(&marks[i]);
}
display(int *m)
{
	printf("%d",*m);
}
```

Pointers in array

```
main()
{
	int marks[]={10,30,20,50,25};
	int *i,*j;
	i=&marks[4]; /* It is equal to (marks+4) */
	j=&marks[0]; /* It is equal to &marks */
	printf("%d %d\n",j-i,*j-*i);
	return 0;
}
```

#Note Elements num[i], *[num+i], *[i+num] and i[num] point to same element.

Passing Entire array to function

```
main()
{
	int marks[]={10,30,20,50,25};
	display(&marks[0],5);
	return 0;
}
display(int *j,int n)
{
	int i;
	for(i=0;i<n;i++)
	{
		printf("Element=%d\n",*j);
		j++; /* Increment pointer to point to next location */
	}
}
```

Two Dimensional Arrays

Syntax: data_type array_name[Row_Size][Column_Size];
data_type array_name[][Coloumn_Size];

Both are syntaxes are legal, but the following syntax is illegal

data_type array_name[Row_Size][];
data_type array_name[][];

E.g. **int xyz[2] [3] = {0,5,7,2,1,3};**
int xyz[] [3] = {0,5,7,2,1,3}; /* This statements is legal */
int xyz[2] [] = {0,5,7,2,1,3}; /* The statements is illegal */
int xyz[] [] = {0,5,7,2,1,3}; /* The statements is also illegal */

we can also initialize a two – dimensional array in the form of a matrix as shown.

```
int xyz[2][3] = {
                    {0,5,7},
                    {2,1,3}
                };
```

If the values are missing in an initializer, they are automatically set to zero.

```
Ex: int xyz[2][3] = {
                        {1,5},                    1 5 0
                        {2}                       2 0 0
                    };
```

E.g.

```
#include<stdio.h>
#include<conio.h>
int main( )
{
        int i,j;
        int a[3][3] = {
                                {1,2,3},
                                {4,5,6},
                                {7,8,9}
                            };
        printf("elements of an array\n");
        for(i=0;i<3; i++)
         {
                for(j=0;j<3;j++)
                        {
                                printf("%d %u\t",a[i][j], &a[i][j]);

                        }
```

```
        }

        getch();
        return 0;
}
```

Memory Map (how it stores in memory)

Element No.	a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]	a[2][0]	a[2][1]	a[2][2]
Value	1	5	17	4	3	26	72	85	91
Address	1024	1026	1028	1030	1032	1034	1036	1038	1040

Passing 2-D array to a function

```
#include<stdio.h>
display(int *q, int row, int col)
{
        int i,j;
        for(i=0;i<row;i++)
        {
                for(j=0;j<col;j++)
                {
                        printf("%d ",*q);
                        q++;
                }

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

show(int *q, int row, int col)
{
        int i,j;
        for(i=0;i<row;i++)
        {
                for(j=0;j<col;j++)
                {
                        printf("%d ",*(q+i*col+j);
                }

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

```
main()
{
	int a[3][4]={1,2,73,48,50,16,7,8,9,14,23,54};
	display(a,3,4);
	show(a,3,4);

}
```

Array of Pointer

```
main()
{
	int *xyz[4]; /* array of integer pointer */
	int i=5,j=7,k=34,l=65,m;
	xyz[0]=&i;
	xyz[1]=&j;
	xyz[2]=&k;
	xyz[3]=&l;
	for(m=0;m<=3;m++)
		printf("%d",*xyz[m]);

}
```

Three-Dimensional Array

```
main()
{
  int i,j,q;
  int a[2][3][4]={
                  {
                        {1,2,73,48},
                        {50,16,7,8},
                        {9,14,23,54}
                  },
                  {
                        {34,672,343,68},
                        {56,46,74,82},
                        {39,174,223,94},
                  }
                };
  for(i=0;i<2;i++)
        {
              for(j=0;j<3;j++)
              {
                    for(q=0;q<4;q++)
                    {
                          printf("%d ",a[i][j][q]);
                    }
                    printf("\n");
```

```
                }
                printf("\n");
        }
        return 0;
}
```