# UNIT: 4
# Array and Pointers

## ❖ Introduction of Array:

**Definition:** Array is a collection of variables of same data type known by same name.

**Types of Arrays:**

- One-dimensional arrays.
- Two-dimensional arrays.
- Multidimensional arrays.

## ❖ Characteristics of an array:

- Array store elements that have same data type.
- Array store elements in subsequent memory location.
- Array size should be mention in the declaration.
- Array name represent the address of starting elements.

## ❖ Declaration and initialization of 1- D (One dimensional array):

**Declaration of array:**

Syntax:
datatype array_name[size];

Data type: it defines the datatype, like int,float,char,double etc.
Array name: it is the name of array.
Size: It represents the size of the array.

Example:

```
int a[5];
float f[5];
```

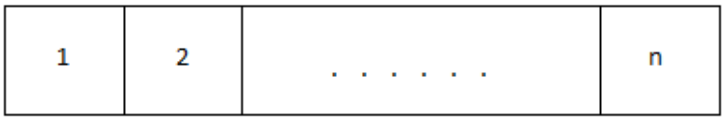



Fig: One Dimensional Array

**Initialization of array:**

**(1) Compile time:**

datatype arrayname[size]={List of value};
int a[5]={10,20,30,40,50};

**(2) Run time:**

```
for(i=0 ; i<5 ; i++)
{
        scanf("%d", &a[i]);
}
```

**Example**: Program to read 5 elements of array and print it.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5],i;
clrscr();
printf("Enter 5 elements of array");
for(i=0;i<5;i++)
{       scanf("%d",&a[i]);
}
printf("The array elements are:\n");

for(i=0;i<5;i++)
{
printf("%d\n",a[i]);
}
getch();
}
```

## ❖ Declaration and initialization of 2- D (Two dimensional array):

**Declaration of array:**

Syntax:

datatype Arrayname [row size] [column size];

Example:

int a[3][3] ;

where, int is datatype, a is a arrayname , row size is 3 and column size is 3.

float f[5][10];

**Initialization of array:**

**(1) Compile time:**

```
int  a[3][3]={ {11,12,13}, {14,15,16}, {17,18,19} } ;
int a[3][3] = {11,12,13,14,15,16,17,18,19};
```

| | col 0 | col 1 | col 2 |
|---|---|---|---|
| row 0 | 11 | 12 | 13 |

VPMP POLYTECHNIC

| | a[0][0] | a[0][1] | a[0][2] |
|---|---|---|---|
| row 1 | 14<br>a[1][0] | 15<br>a[1][1] | 16<br>a[1][1] |
| row 2 | 17<br>a[2][0] | 18<br>a[2][1] | 19<br>a[2][2] |

**(In memory Representation of array)**

**(2) Run time:**

```
for(i=0 ; i<3 ; i++)
{
for(j=0 ; j<3 ; j++)
        {
        scanf(" %d ", &a[i][j] );
        }
 }
```

**Example**: Program to read 3X3 matrix (2D array elements) and print it.

```
#include<stdio.h>
void main()
{
   int i,j,a[3][3];
   printf("Enter elements of 3X3 matrix\n");
   for(i=0;i<3;i++)
   {
      for(j=0;j<3;j++)
      {
         scanf("%d",&a[i][j]);
      }
   }
printf("The elements of 3X3 matrix\n");
   for(i=0;i<3;i++)
   {
      for(j=0;j<3;j++)
      {
         printf("%d\t",a[i][j]);
      }
printf( "\n" );
   }
```

VPMP POLYTECHNIC

## ❖ Introduction to a String:

**String:**

A string is a sequence of zero or more characters followed by a NULL '\0' character.
String is always terminating with NULL '\0' character.

### Declaration and initialization of string:

Syntax:

char stringname[size];
where size is the number of characters.

Example:

char city[10] ;

### Initialization of string variable:

**(1) Compile time:**

```
char city[9]= " NEW YORK";
```

**(2) Run time:**

```
char city[9];
scanf("%s", city);
```

OR

```
char city[9];
gets(ciy);
```

## ❖ gets() and puts() :

### gets():

This function read a string from a user. It is defined in the <stdio.h> header file of C.

### puts() :

This function prints a string on the console screen. It is also defined in the <stdio.h> header file of C.

**Example**: Program to read and print a string using gets() and puts().

```
void main()
{
char day[10];
printf("Enter day: \n");
gets(day);

printf("Today is: ");
puts(day);
}
```

Output:

VPMP POLYTECHNIC

```
Enter day: TUESDAY
Today is: TUESDAY
```

## ❖ Pointer

## ❖ Introduction to Pointer:

Pointer is a variable which store the address of another variable.

**Advantage of pointer:**

- Pointer reduces the code and improves the performance.
- We can return multiple values from a function using the pointer.
- It makes you able to access any memory location in the computer's memory.

**Characteristics of Pointers:**

- Pointer is a variable which can hold the address of another variable.
- A pointer is a derived data type.
- It contains memory addresses as their values.
- If a C pointer is assigned to the null value, it points nothing.
- The asterisk symbol, * is used to retrieve the value of the variable
- & ampersand symbol is used for retrieving the address of a variable.

## ❖ declaration and initialization of pointer :

**Declaration of pointer**

Syntax:

Data type *pointer-name

Example:

```
int *p;
```

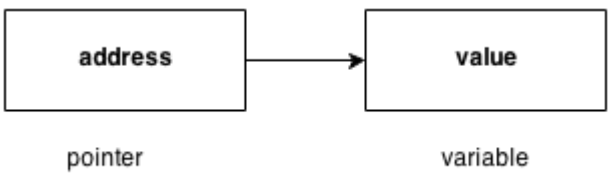
**Initialization of pointer:**

Syntax:

Pointer variable= &variable-name;

Example:

```
int price;
int *p;
p=&price;
```




VPMP POLYTECHNIC

Example:

```
#include<stdio.h>
Void main(){
int number=50;
int *p;
p=&number;//stores the address of number variable
printf("Address of p variable is %x \n",p); // p contains the address of the number
printf("Value of p variable is %d \n",*p); // * is used to dereference a pointer therefore *p means get the value stored at the address contained by p.
}
```

## ❖ Types of pointer : void and null

## void pointer

Syntax:

```
void *variable name
```

- Void pointer is a pointer which has no specified data type.
- It is also known as generic pointer.
- Void pointer can be pointed to any data type.
- When void pointer is declared two bytes of memory is assign to it.
- It is used when return type or parameter is unknown.
- void pointers that have addresses, can be further typecast into other data types very easily.

Example:

```
include<stdio.h>
void  main()
{
int a = 10;
void *p;
p = &a;
printf("%d", *(int *)p);
}
```

## NULL Pointer:

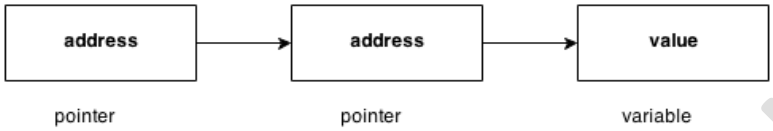
- A pointer that is not assigned any value but NULL is known as the NULL pointer.
- If you don't have any address to be specified in the pointer at the time of declaration, you can assign NULL value. It will provide a better approach.

```
int *p=NULL;
```

VPMP POLYTECHNIC

## pointer to pointer:

- Pointer variable may pointes to another pointer variable.
- It is also known as double pointer.
- The first pointer is used to store the address of a variable whereas the second pointer is used to store the address of the first pointer.




Syntax:
Data type **pointer variable name;
Example:

```
int **p; // pointer to a pointer which is pointing to an integer.
```

Example:

```
int n,*p1,**p2;
n=10;
p1=&n;
p2=&p1;
```

Example:

```
#include<stdio.h>
void main ()
{
   int a = 10;
   int *p;
   int **p2;
   p = &a; // pointer p is pointing to the address of a
   p2 = &p; // pointer p2 is a double pointer pointing to the address of pointer p
   printf("address of a: %x\n",p); // Address of a will be printed
   printf("address of p: %x\n",p2); // Address of p will be printed
   printf("value stored at p: %d\n",*p); // value stoted at the address contained by p i.e. 10 will be printed
   printf("value stored at p2: %d\n",**pp); // value stored at the address contained by pointer stored at p2
}
```

VPMP POLYTECHNIC