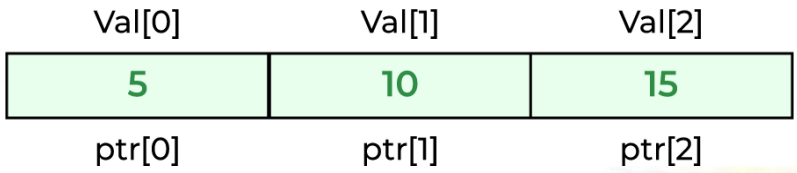
4.4.POINTERS AND ONE DIMENSIONAL ARRAYS

In C programming language, pointers and arrays are closely related. An array name acts like a pointer constant. The value of this pointer constant is the address of the first element. For example, if we have an array named val then **val** and **&val[0]** can be used interchangeably.

If we assign this value to a non-constant pointer of the same type, then we can access the elements of the array using this pointer.

Example 1: Accessing Array Elements using Pointer with Array Subscript



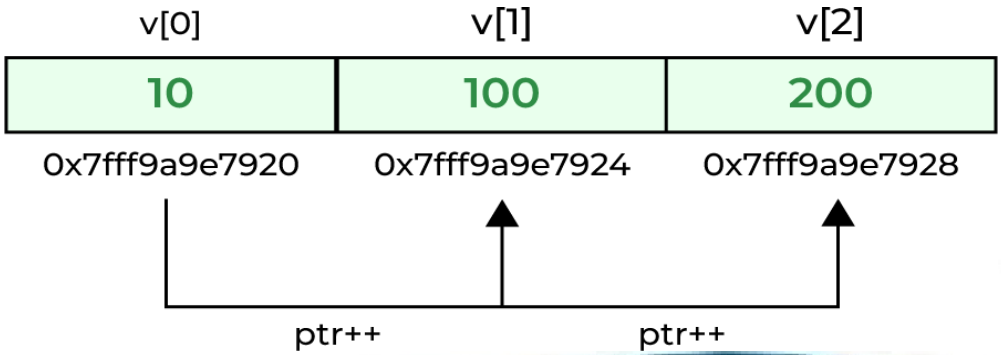

```
// C Program to access array elements using pointer
#include <stdio.h>
void p_array()
{
   int val[3] = {5,10,15};
   int* ptr;
   ptr = val;
   printf("Elements of the array are: ");
  printf("%d, %d, %d", ptr[0], ptr[1], ptr[2]);
  return;
}
int main()
{
   p_array();
   return 0;
}
```

Output

```
Elements of the array are: 5, 10, 15
```

Not only that, as the array elements are stored continuously, we can pointer arithmetic operations such as increment, decrement, addition, and subtraction of integers on pointer to move between array elements.

Example 2: Accessing Array Elements using Pointer Arithmetic




```
// C Program to access array elements using pointers
#include <stdio.h>
int main()
{
   int arr[5] = { 1, 2, 3, 4, 5 };
   int* ptr_arr = arr;
   for (int i = 0; i < 5; i++)
      { printf("%d ", *ptr_arr++);
   }
   return 0;
}
```

Output

```
1 2 3 4 5
```

Explain pointers and one-dimensional array in C language

Pointers and one-dimensional arrays

The compiler allocates Continuous memory locations for all the elements of the array.

The base address = first element address (index 0) of the array.

- For Example − int a [5] = {10, 20,30,40,50};

Elements

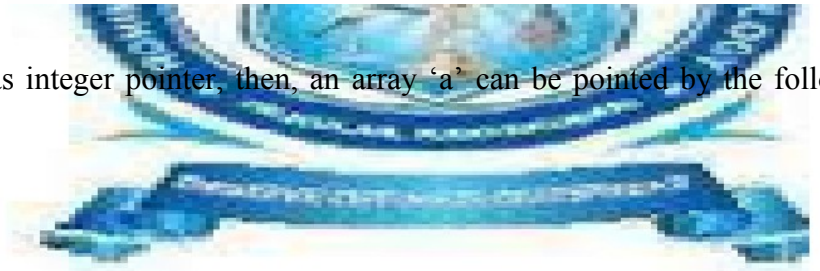
The five elements are stored as follows −

Elements	a[0]	a[1]	a[2]	a[3]	a[4]
Value	10	20	30	40	50
Address	1000	1004	1008	1012	1016

base address

a= &a[0]=1000

If ‘p’ is declared as integer pointer, then, an array ‘a’ can be pointed by the following assignment −



p = a;

(or) p = &a[0];

Every value of 'a' is accessed by using p++ to move from one element to another element. When a pointer is incremented, its value is increased by the size of the datatype that it points to. This length is called the "scale factor".

The relationship between ‘p’ and 'a' is explained below

− P = &a[0] = 1000

P+1 = &a[1] = 1004

P+2 = &a[2] = 1008

P+3 = &a[3] = 1012

P+4 = &a[4] = 1016

Address of an element is calculated using its index and the scale factor of the datatype. An example to explain this is given herewith.

Address of a[3] = base address + (3* scale factor of int)

= 1000 + (3*4)

= 1000 +12

= 1012

Pointers can be used to access array elements instead of using array indexing.

*(p+3) gives the value of a[3].

a[i] = *(p+i)

Example program

Following is the C program for pointers and one-dimensional arrays −

```
#include<stdio.h>
main ( ){
  int
  a[5];
  int *p,i;
  printf ("Enter 5 lements");
  for (i=0; i<5; i++)
    scanf ("%d", &a[i]);
  p = &a[0];
  printf ("Elements of the array are");
  for (i=0; i<5; i++)
    printf("%d", *(p+i));
```

Output

When the above program is executed, it produces the following result −

Enter 5 elements: 10 20 30 40 50

Elements of the array are : 10 20 30 40 50