Pointer Expressions and Pointer Arithmetic

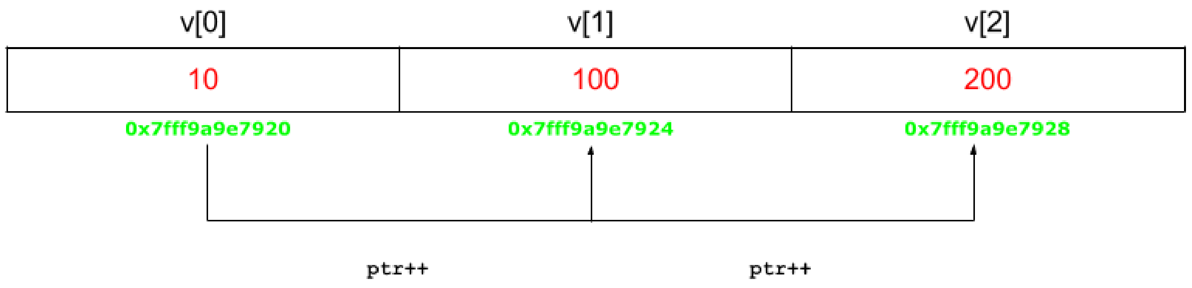
A limited set of arithmetic operations can be performed on pointers. A pointer may be:

- incremented (++)
- decremented (—)
- an integer may be added to a pointer (+ or +=)
- an integer may be subtracted from a pointer (– or -=)

Pointer arithmetic is meaningless unless performed on an array.
Note : Pointers contain addresses. Adding two addresses makes no sense, because there is no idea what it would point to. Subtracting two addresses lets you compute the offset between these two addresses.

```
// C++ program to illustrate Pointer Arithmetic
// in C/C++
#include <bits/stdc++.h>

// Driver program
int main()
{
    // Declare an array
    int v[3] = {10, 100, 200};

    // Declare pointer variable
    int *ptr;

    // Assign the address of v[0] to ptr
    ptr = v;

    for (int i = 0; i < 3; i++)
    {
        printf("Value of *ptr = %d\n", *ptr);
        printf("Value of ptr = %p\n\n", ptr);

        // Increment pointer ptr by 1
        ptr++;
    }
}
```

```
Output:Value of *ptr = 10
Value of ptr = 0x7ffcae30c710

Value of *ptr = 100
Value of ptr = 0x7ffcae30c714

Value of *ptr = 200
Value of ptr = 0x7ffcae30c718
```

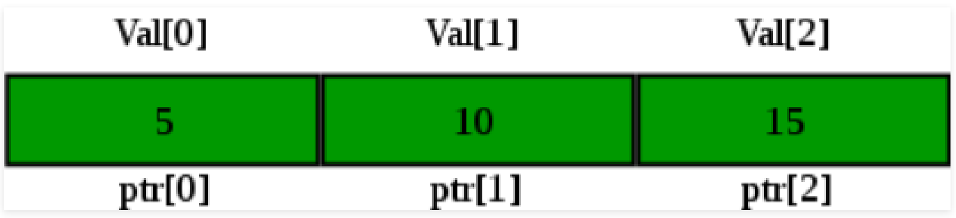


Array Name as Pointers

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.

```
// C++ program to illustrate Array Name as Pointers in C++
#include <bits/stdc++.h>
using namespace std;

void geeks()
{
    // Declare an array
    int val[3] = { 5, 10, 20 };

    // Declare pointer variable
    int *ptr;

    // Assign address of val[0] to ptr.
    // We can use ptr=&val[0];(both are same)
    ptr = val ;
    cout << "Elements of the array are: ";
    cout << ptr[0] << " " << ptr[1] << " " << ptr[2];

    return;
}


// Driver program
int main()
{
    geeks();
    return 0;
}
```

```
Output:
Elements of the array are: 5 10 20
```



Pointers and Multidimensional Arrays

Consider pointer notation for the two-dimensional numeric arrays. consider the following declaration

```
int nums[2][3]  =  { {16, 18, 20}, {25, 26, 27} };
```

In general, nums[i][j] is equivalent to *(*(nums+i)+j)

POINTER NOTATION	ARRAY NOTATION	VALUE
*(*nums)	nums[0][0]	16
*(*nums + 1)	nums[0][1]	18
*(*nums + 2)	nums[0][2]	20
((nums + 1))	nums[1][0]	25
((nums + 1) + 1)	nums[1][1]	26
((nums + 1) + 2)	nums[1][2]	27