# Chapter - 9 : Pointers

## Pointer Arithmetic

- A pointer variable stores the address of location and it is numeric.
- Operators allowed on pointers are ++ , -- , + and -.
- Unary Operator effects on different data types.

| Data Type | Initial Address | Operation | Address after Operation | Required Bytes |
|---|---|---|---|---|
| int | 4000 | ++ | 4002 | 2 |
| int | 4000 | -- | 3998 | 2 |
| char | 4000 | ++ | 4001 | 1 |
| char | 4000 | -- | 3999 | 1 |
| float | 4000 | ++ | 4004 | 4 |
| float | 4000 | -- | 3996 | 4 |
| long | 4000 | ++ | 4004 | 4 |
| long | 4000 | -- | 3996 | 4 |

- Arithmetic Operator effects on different data types

| Expression | Result |
|---|---|
| Address + Number | Address |
| Address – Number | Address |
| Address – Address | Number |
| Address + Address | Illegal |

**Example -1**

```
// Program to print the value of an array using the pointer arithmetic

#include <stdio.h>

void main ()
{
        int  x[ ] = {10, 100, 200};
        int  i, *ptr;

        ptr = x;      /* Assign array address to pointer */

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

          ptr++;      /* move to the next location */

         }
 }
```

**Example -2**

**// Program to print the reverse of an array using the pointer arithmetic**

```
#include <stdio.h>

void main ()
{
        int  x[ ] = {10, 100, 200};
        int  i, *ptr;

        ptr = &x[2];      /* Assign array address to pointer */

        for ( i = 2; i >= 0; i--)
        {
          printf("Value of var[%d] = %d\n", i, *ptr );

          ptr--;      /* move to the previous location */

         }
 }
```

| **X[ ] value** | 10 | 100 | 200 |
|---|---|---|---|
| **Address** | 4000 | 4002 | 4004 |

ptr=x;  it sets the **ptr** pointer to 4000.

ptr++  is equal to ptr + 1  ( 4000  +  2 = 4002)

ptr-- is equal to ptr – 1 ( 4002 – 2 = 4000 )

## Array and Pointer

- An array name without brackets is a pointer to the array′s first element.
- If a programmer has declared an array data[ ], data is the address of the first array element.
- For example, the following code initializes the pointer variable p_array with the address of the first element of array[ ].

  **int array[10], *p_array;**

  **p_array=arrary;**
- **p_array** is a pointer variable, it can be modified to point elsewhere.
- Unlike array, p_array isn′t locked into pointing at the first element of array[ ].
- Example, it could be pointed at other elements of array[ ].
- Array name without subscript points to first element in an array.

## Example : One Dimensional Array

int a[10];

Here a , a[0] both are identical.

a[i] = *(a + i)  or *(i + a)

To access the a[4],  we can write *(a+4).

## Example : Two Dimensional Array

int a[10][10];

Here a and a[0][0] both are identical.

a[i][j] = *(a[i]+j)  or *(j + a[i])

To access the a[4][3],  we can write *(a[4]+3).

## Strings and Pointers

- To allocate and initialize a string is to declare an array of type char as follows:
  **char message[ ] = "This is the message.";**
- One could accomplish the same thing by declaring a pointer to type char:
  **char *message = "This is the message.";**

## Example

```
#include <stdio.h>
void main ()
{
  double  balance[5] = {1000.0, 2.0,10.0, 17.0, 50.0};
  double *ptr;

  ptr = balance;  /* Assign array address to pointer */

  for ( i = 0; i < 3; i++)
  {       printf("Value of var[%d] = %d\n", i, *(ptr+i) );   }
  for ( i = 0; i < 3; i++)
  {      printf("Value of var[%d] = %d\n", i, *(balance+i) );    }
}
```

## Example

```
#include <stdio.h>
void main ()
{
  char  arr[5][20] = {"C","C++","Java","Python","Android"};
  char **ptr;

  ptr = arr;     /* Assign array address to pointer */

  for ( i = 0; i < 5; i++)
  {       printf("Value of var[%d] = %d\n", i, *ptr[i]);    }

}
```