C++ POINTERS VS ARRAYS

 Copyright © tutorialspoint.com

Pointers and arrays are strongly related. In fact, pointers and arrays are interchangeable in many cases. For example, a pointer that points to the beginning of an array can access that array by using either pointer arithmetic or array-style indexing. Consider the following program:

```
#include <iostream>

using namespace std;
const int MAX = 3;

int main ()
{
   int  var[MAX] = {10, 100, 200};
   int  *ptr;

   // let us have array address in pointer.
   ptr = var;
   for (int i = 0; i < MAX; i++)
   {
      cout << "Address of var[" << i << "] = ";
      cout << ptr << endl;

      cout << "Value of var[" << i << "] = ";
      cout << *ptr << endl;

      // point to the next location
      ptr++;
   }
   return 0;
}
```

When the above code is compiled and executed, it produces result something as follows:

```
Address of var[0] = 0xbfa088b0
Value of var[0] = 10
Address of var[1] = 0xbfa088b4
Value of var[1] = 100
Address of var[2] = 0xbfa088b8
Value of var[2] = 200
```

However, pointers and arrays are not completely interchangeable. For example, consider the following program:

```
#include <iostream>

using namespace std;
const int MAX = 3;

int main ()
{
   int  var[MAX] = {10, 100, 200};

   for (int i = 0; i < MAX; i++)
   {
      *var = i;    // This is a correct syntax
      var++;       // This is incorrect.
   }
   return 0;
}
```

It is perfectly acceptable to apply the pointer operator * to var but it is illegal to modify var value. The reason for this is that var is a constant that points to the beginning of an array and can not be

used as l-value.

Because an array name generates a pointer constant, it can still be used in pointer-style expressions, as long as it is not modified. For example, the following is a valid statement that assigns var[2] the value 500:

```
*(var + 2) = 500;
```

Above statement is valid and will compile successfully because var is not changed.