PASCAL - POINTER ARITHMETIC

As explained in main chapter, Pascal pointer is an address, which is a numerical value stored in a word. Therefore, you can perform arithmetic operations on a pointer just as you can on a numeric value. There are four arithmetic operators that can be used on pointers: increment, decrement, +, and -.

To understand pointer arithmetic, let us consider that ptr is an integer pointer, which points to the address 1000. Assuming 32-bit integers, let us perform the increment operation on the pointer −

```
Inc(ptr);
```

Now, after the above operation, the **ptr** will point to the location 1004 because each time **ptr** is incremented, it will point to the next integer location, which is 4 bytes next to the current location. This operation will move the pointer to next memory location without impacting actual value at the memory location. If **ptr** points to a character, whose address is 1000, then above operation will point to the location 1001 because next character will be available at 1001.

Incrementing a Pointer

We prefer using a pointer in our program instead of an array because the variable pointer can be incremented, unlike the array name, which cannot be incremented because it is a constant pointer. The following program increments the variable pointer to access each succeeding element of the array −

```
program exPointers;
const MAX = 3;
var
   arr: array [1..MAX] of integer = (10, 100, 200);
   i: integer;
   iptr: ^integer;
   y: ^word;

begin
   (* let us have array address in pointer *)
   iptr := @arr[1];

   for  i := 1 to MAX do
   begin
      y:= addr(iptr);
      writeln('Address of arr[', i, '] = ' , y^ );
      writeln(' Value of arr[', i, '] = ' , iptr^ );

      (* move to the next location *)
      inc(iptr);
   end;
end.
```

When the above code is compiled and executed, it produces the following result −

```
Address of arr[1] = 32880
Value of arr[1] = 10
Address of arr[2] = 32882
Value of arr[2] = 100
Address of arr[3] = 32884
Value of arr[3] = 200
```

Decrementing a Pointer

The same considerations apply to decrementing a pointer, which decreases its value by the number of bytes of its data type as shown below −

```
program exPointers;
const MAX = 3;
var
   arr: array [1..MAX] of integer = (10, 100, 200);
   i: integer;
   iptr: ^integer;
   y: ^word;

begin
   (* let us have array address in pointer *)
   iptr := @arr[MAX];

   for  i := MAX downto 1 do
   begin
      y:= addr(iptr);
      writeln('Address of arr[', i, '] = ' , y^ );
      writeln(' Value of arr[', i, '] = ' , iptr^ );

      (* move to the next location *)
      dec(iptr);
   end;
end.
```

When the above code is compiled and executed, it produces the following result −

```
Address of arr[3] = 32884
Value of arrr[3] = 200
Address of arr[2] = 32882
Value of arr[2] = 100
Address of arr[1] = 32880
Value of arr[1] = 10
```

Pointer Comparisons

Pointers may be compared by using relational operators, such as =, <, and >. If p1 and p2 point to variables that are related to each other, such as elements of the same array, then p1 and p2 can be meaningfully compared.

The following program modifies the previous example one by incrementing the variable pointer so long as the address to which it points is either less than or equal to the address of the last element of the array, which is @arr[MAX] −

```
program exPointers;
const MAX = 3;
var
   arr: array [1..MAX] of integer = (10, 100, 200);
   i: integer;
   iptr: ^integer;
   y: ^word;

begin
   i:=1;

   (* let us have array address in pointer *)
   iptr := @arr[1];

   while (iptr <= @arr[MAX]) do
   begin
      y:= addr(iptr);
      writeln('Address of arr[', i, '] = ' , y^ );
      writeln(' Value of arr[', i, '] = ' , iptr^ );

      (* move to the next location *)
      inc(iptr);
      i := i+1;
   end;
end.
```

When the above code is compiled and executed, it produces the following result −

```
Address of arr[1] = 32880
Value of arr[1] = 10
Address of arr[2] = 32882
Value of arr[2] = 100
Address of arr[3] = 32884
Value of arr[3] = 200
```