PASCAL - POINTERS

Pointers in Pascal are easy and fun to learn. Some Pascal programming tasks are performed more easily with pointers, and other tasks, such as dynamic memory allocation, cannot be performed without using pointers. So it becomes necessary to learn pointers to become a perfect Pascal programmer. Let's start learning them in simple and easy steps.

As you know, every variable is a memory location and every memory location has its address defined which can be accessed using the name of the pointer variable, which denotes an address in memory.

What Are Pointers?

A pointer is a dynamic variable, whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before you can use it to store any variable address. The general form of a pointer variable declaration is −

```
type
   ptr-identifier = ^base-variable-type;
```

The pointer type is defined by prefixing the up-arrow of caret symbol ^ with the base type. The base-type defines the types of the data items. Once a pointer variable is defined to be of certain type, it can point data items of that type only. Once a pointer type has been defined, we can use the **var** declaration to declare pointer variables.

```
var
   p1, p2, ... : ptr-identifier;
```

Following are some valid pointer declarations −

```
type
   Rptr = ^real;
   Cptr = ^char;
   Bptr = ^ Boolean;
   Aptr = ^array[1..5] of real;
   date-ptr = ^ date;
      Date = record
         Day: 1..31;
         Month: 1..12;
         Year: 1900..3000;
      End;
var
   a, b : Rptr;
   d: date-ptr;
```

The pointer variables are dereferenced by using the same caret symbol ^. For example, the associated variable referred by a pointer *rptr*, is *rptr*^. It can be accessed as −

```
rptr^ := 234.56;
```

The following example will illustrate this concept −

```
program exPointers;
var
   number: integer;
   iptr: ^integer;

begin
   number := 100;
   writeln('Number is: ', number);
```

```
   iptr := @number;
   writeln('iptr points to a value: ', iptr^);

   iptr^ := 200;
   writeln('Number is: ', number);
   writeln('iptr points to a value: ', iptr^);
end.
```

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

```
Number is: 100
iptr points to a value: 100
Number is: 200
iptr points to a value: 200
```

Printing a Memory Address in Pascal

In Pascal, we can assign the address of a variable to a pointer variable using the address operator @. We use this pointer to manipulate and access the data item. However, if for some reason, we need to work with the memory address itself, we need to store it in a word type variable.

Let us extend the above example to print the memory address stored in the pointer *iptr* −

```
program exPointers;
var
   number: integer;
   iptr: ^integer;
   y: ^word;

begin
   number := 100;
   writeln('Number is: ', number);
   iptr := @number;
   writeln('iptr points to a value: ', iptr^);

   iptr^ := 200;
   writeln('Number is: ', number);
   writeln('iptr points to a value: ', iptr^);
   y := addr(iptr);
   writeln(y^);
end.
```

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

```
Number is: 100
iptr points to a value: 100
Number is: 200
iptr points to a value: 200
36864
```

NIL Pointers

It is always a good practice to assign a **NIL** value to a pointer variable in case you do not have exact address to be assigned. This is done at the time of variable declaration. A pointer that is assigned **NIL** points to nowhere. Consider the following program −

```
program exPointers;
var
   number: integer;
   iptr: ^integer;
   y: ^word;

begin
   iptr := nil;
   y := addr(iptr);
```

```
   writeln('the vaule of iptr is ', y^);
end.
```

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

```
The value of ptr is 0
```

To check for a **nil** pointer you can use an if statement as follows −

```
if(ptr <> nill )then     (* succeeds if p is not null *)
if(ptr = nill)then    (* succeeds if p is null *)
```

Pascal Pointers in Detail

Pointers have many but easy concepts and they are very important to Pascal programming. There are following few important pointer concepts, which should be clear to a Pascal programmer −

Concept	Description
Pascal - Pointer arithmetic	There are four arithmetic operators that can be used on pointers: increment,decrement, +, -
Pascal - Array of pointers	You can define arrays to hold a number of pointers.
Pascal - Pointer to pointer	Pascal allows you to have pointer on a pointer and so on.
Passing pointers to subprograms in Pascal	Passing an argument by reference or by address both enable the passed argument to be changed in the calling subprogram by the called subprogram.
Return pointer from subprograms in Pascal	Pascal allows a subprogram to return a pointer.