PASCAL - VARIABLE TYPES

A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in Pascal has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Pascal is **not case-sensitive**, so uppercase and lowercase letters mean same here. Based on the basic types explained in previous chapter, there will be following basic variable types −

Basic Variables in Pascal

Type	Description
Character	Typically a single octet *onebyte*. This is an integer type.
Integer	The most natural size of integer for the machine.
Real	A single-precision floating point value.
Boolean	Specifies true or false logical values. This is also an integer type.
Enumerated	Specifies a user-defined list.
Subrange	Represents variables, whose values lie within a range.
String	Stores an array of characters.

Pascal programming language also allows defining various other types of variables, which we will cover in subsequent chapters like Pointer, Array, Records, Sets, and Files, etc. For this chapter, let us study only basic variable types.

Variable Declaration in Pascal

All variables must be declared before we use them in Pascal program. All variable declarations are followed by the *var* keyword. A declaration specifies a list of variables, followed by a colon : and the type. Syntax of variable declaration is −

```
var
variable_list : type;
```

Here, type must be a valid Pascal data type including character, integer, real, boolean, or any user-defined data type, etc., and variable_list may consist of one or more identifier names separated by commas. Some valid variable declarations are shown here −

```
var
age, weekdays : integer;
taxrate, net_income: real;
choice, isready: boolean;
initials, grade: char;
name, surname : string;
```

In the previous tutorial, we have discussed that Pascal allows declaring a type. A type can be identified by a name or identifier. This type can be used to define variables of that type. For example,

```
type
```

```
days, age = integer;
yes, true = boolean;
name, city = string;
fees, expenses = real;
```

Now, the types so defined can be used in variable declarations −

```
var
weekdays, holidays : days;
choice: yes;
student_name, emp_name : name;
capital: city;
cost: expenses;
```

Please note the difference between *type* declaration and *var* declaration. Type declaration indicates the category or class of the types such as integer, real, etc., whereas the variable specification indicates the type of values a variable may take. You can compare *type* declaration in Pascal with *typedef* in C. Most importantly, the variable name refers to the memory location where the value of the variable is going to be stored. This is not so with the type declaration.

Variable Initialization in Pascal

Variables are assigned a value with a colon and the equal sign, followed by a constant expression. The general form of assigning a value is −

```
variable_name := value;
```

By default, variables in Pascal are not initialized with zero. They may contain rubbish values. So it is a better practice to initialize variables in a program. Variables can be initialized *assignedaninitialvalue* in their declaration. The initialization is followed by the **var** keyword and the syntax of initialization is as follows −

```
var
variable_name : type = value;
```

Some examples are −

```
age: integer = 15;
taxrate: real = 0.5;
grade: char = 'A';
name: string = 'John Smith';
```

Let us look at an example, which makes use of various types of variables discussed so far −

```
program Greetings;
const
message = ' Welcome to the world of Pascal ';

type
name = string;
var
firstname, surname: name;

begin
   writeln('Please enter your first name: ');
   readln(firstname);

   writeln('Please enter your surname: ');
   readln(surname);

   writeln;
   writeln(message, ' ', firstname, ' ', surname);
end.
```

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

```
Please enter your first name:
John
Please enter your surname:
Smith
Welcome to the world of Pascal John Smith
```

Enumerated Variables

You have seen how to use simple variable types like integer, real and boolean. Now, let's see variables of enumerated type, which can be defined as −

```
var
var1, var2, ...  : enum-identifier;
```

When you have declared an enumerated type, you can declare variables of that type. For example,

```
type
months = (January, February, March, April, May, June, July, August, September, October,
November, December);
Var
m: months;
...
M := January;
```

The following example illustrates the concept −

```
program exEnumeration;
type
beverage = (coffee, tea, milk, water, coke, limejuice);

var
drink:beverage;

begin
   writeln('Which drink do you want?');
   drink := limejuice;

   writeln('You can drink ', drink);
end.
```

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

```
Which drink do you want?
You can drink limejuice
```

Subrange Variables

Subrange variables are declared as −

```
var
subrange-name : lowerlim ... uperlim;
```

Examples of subrange variables are −

```
var
marks: 1 ... 100;
grade: 'A' ... 'E';
age: 1 ... 25;
```

The following program illustrates the concept −

```
program exSubrange;
var
marks: 1 .. 100;
grade: 'A' .. 'E';

begin
   writeln( 'Enter your marks(1 - 100): ');
   readln(marks);

   writeln( 'Enter your grade(A - E): ');
   readln(grade);

   writeln('Marks: ' , marks, ' Grade: ', grade);
end.
```

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

```
Enter your marks(1 - 100):
100
Enter your grade(A - E):
A
Marks: 100 Grade: A
```