Data types of an entity indicates the meaning, constraints, possible values, operations, functions and mode of storage associated with it.

Integer, real, Boolean and character types are referred as standard data types. Data types can be categorized as scalar, pointer and structured data types. Examples of scalar data types are integer, real, Boolean, character, subrange and enumerated. Structured data types are made of the scalar types; for example, arrays, records, files and sets. We will discuss the pointer data types later.

**Pascal Data Types**

Pascal data types can be summarized as below in the following diagram –

Type Declarations

The type declaration is used to declare the data type of an identifier. Syntax of type declaration is –

```
type-identifier-1, type-identifier-2 = type-specifier;
```

For example, the following declaration defines the variables days and age as integer type, yes and true as Boolean type, name and city as string type, fees and expenses as real type.

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

**Integer Types**

Following table gives you details about standard integer types with its storage sizes and value
ranges used in Object Pascal –

<table>
<thead>
<tr>
<th>Type</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>-2147483648</td>
<td>2147483647</td>
<td>signed 32-bit</td>
</tr>
<tr>
<td>Cardinal</td>
<td>0</td>
<td>4294967295</td>
<td>unsigned 32-bit</td>
</tr>
<tr>
<td>Shortint</td>
<td>-128</td>
<td>127</td>
<td>signed 8-bit</td>
</tr>
<tr>
<td>Smallint</td>
<td>-32768</td>
<td>32767</td>
<td>signed 16-bit</td>
</tr>
<tr>
<td>Longint</td>
<td>-2147483648</td>
<td>2147483647</td>
<td>signed 32-bit</td>
</tr>
<tr>
<td>Int64</td>
<td>-2^63</td>
<td>2^63 - 1</td>
<td>signed 64-bit</td>
</tr>
<tr>
<td>Byte</td>
<td>0</td>
<td>255</td>
<td>unsigned 8-bit</td>
</tr>
<tr>
<td>Word</td>
<td>0</td>
<td>65535</td>
<td>unsigned 16-bit</td>
</tr>
<tr>
<td>Longword</td>
<td>0</td>
<td>4294967295</td>
<td>unsigned 32-bit</td>
</tr>
</tbody>
</table>

**Constants**

Use of constants makes a program more readable and helps to keep special quantities at one place in the beginning of the program. Pascal allows numerical, logical, string and character constants. Constants can be declared in the declaration part of the program by specifying the `const` declaration.

Syntax of constant type declaration is follows –

```pascal
const
  Identifier = constant_value;
```

Following are some examples of constant declarations –

```pascal
VELOCITY_LIGHT = 3.0E+10;
PIE = 3.141592;
NAME = 'Stuart Little';
CHOICE = yes;
OPERATOR = '+';
```

All constant declarations must be given before the variable declaration.

**Enumerated types**

Enumerated data types are user-defined data types. They allow values to be specified in a list. Only `assignment` operators and `relational` operators are permitted on enumerated data type.

Enumerated data types can be declared as follows –

```pascal
type
  enum-identifier = (item1, item2, item3, ... )
```

Following are some examples of enumerated type declarations –

```pascal
type
  SUMMER = (April, May, June, July, September);
  COLORS = (Red, Green, Blue, Yellow, Magenta, Cyan, Black, White);
  TRANSPORT = (Bus, Train, Airplane, Ship);
```

The order in which the items are listed in the domain of an enumerated type defines the order of the items. For example, in the enumerated type SUMMER, April comes before May, May comes before June, and so on. The domain of enumerated type identifiers cannot consist of numeric or
character constants.

**Subrange Types**

Subrange types allow a variable to assume values that lie within a certain range. For example, if the *age* of voters should lie between 18 to 100 years, a variable named `age` could be declared as:

```markdown
var
age: 18 ... 100;
```

We will look at variable declaration in detail in the next section. You can also define a subrange type using the type declaration. Syntax for declaring a subrange type is as follows:

```markdown
type subrange-identifier = lower-limit ... upper-limit;
```

Following are some examples of subrange type declarations:

```markdown
const
P = 18;
Q = 90;
type
Number = 1 ... 100;
Value = P ... Q;
```

Subrange types can be created from a subset of an already defined enumerated type, for example:

```markdown
type
months = (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec);
Summer = Apr ... Aug;
Winter = Oct ... Dec;
```