

Chapter 3

□ ELEMENTARY DATA TYPES

A program manipulates *data*, which is stored in the memory of the computer. Individual items of data are stored in ‘boxes’ in memory which are given *variable names*. These names are *identifiers* which allow us to refer to memory boxes by name rather than use the cumbersome *machine address*.

Each variable name (called simply a *variable* from now on) has a *type* associated with it. A type defines

1. A set of values
2. A set of operations that can be performed on the values.

3.1 Built-In Data Types

Pascal has 4 built-in scalar types and allows the programmer to declare new types in terms of previously-defined types.

INTEGER Type This type represents the integer numbers in the range $-Maxint(=-2^{15})$ to $+Maxint(=+2^{15})$ along with the operations :

1. + add
2. - subtract
3. * multiply
4. *div* division with truncation
5. *mod* remainder after division

In Turbo Pascal and integer is represented by 2 Bytes = 16 bits with one bit for the sign and 15 bits for the magnitude. A block of 15 bits can represent 2^{15} different magnitudes. Hence $Maxint = 32767$. *Maxint* is a built-in integer constant.

The first 3 operations are the familiar ones; the last two are not so familiar. Here are some examples :

$$7 \text{ DIV } 3 = 2$$

$$8 \text{ DIV } 9 = 0$$

$$-24 \text{ DIV } 5 = -4$$

$$56 \text{ DIV } 76 = 0$$

$$32 \text{ MOD } 7 = 4$$

$$6 \text{ MOD } 3 = 0$$

$$75 \text{ MOD } 50 = 25$$

We do not consider the MOD operation for negative integers.

Along with the integer operations Pascal has the following integer functions :

1. $Abs(x) = |x|$,
2. $Sqr(x) = x^2$.
3. $Succ(x) = x + 1$,
4. $Pred(x) = x - 1$.

The square root function is not defined for integers because this could result in a *non-integer*

REAL Type This type represents numbers of the form

$$\pm.f \times b^e$$

with magnitudes in the range 10^{-38} to 10^{+38} along with the operations :

1. + add
2. - subtract
3. * multiply
4. / division

In Turbo Pascal a real is represented by 6 Bytes and has 11 significant decimal digits in the fractional part f .

Pascal has the following functions for reals :

1. $Abs(x)$
2. $Sqr(x)$
3. $Sqrt(x)$
4. $\ln(x) = \log_e x$
5. $\exp(x) = e^x$
6. $\sin(x)$
7. $\cos(x)$

8. $\arctan(x)$

BOOLEAN Type This type represents the values *true* and *false* along with the operations :

1. **and**
2. **or**
3. **not**

These operators are defined as follows :

p	q	p and q	p or q	not p
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

Boolean values occur when we use *relational* operators. These are

1. =
2. <>
3. <
4. <=
5. >
6. >=

These relational operators can be applied to Integers, Reals, Characters and Strings. For example

$11 < 2$ is false

$-21.4 > -32.0$ is true

'A' > 'B' is false

'Jones' < 'Walsh' is true

These relational expressions can be combined using the Boolean operators as follows

$11 < 2$ **and** $-21.4 > -32.0$ is false

CHAR Type This type represents the 128 characters in the ASCII character set. This set contains the upper and lower case alphabetic characters, the digits 0 to 9, punctuation marks, special symbols and control characters. The characters have the following operations :

1. $Chr(i)$ = character corresponding to i in the ASCII table.
2. $Ord(c)$ = number corresponding to c in the ASCII table.

3. $Succ(c)$ = character succeeding c in ASCII order.
4. $Pred(c)$ = character preceding c in ASCII order.

Characters may be compared using the relational operators. For example, if `ch1` and `ch2` are character variables then we may perform tests such as `ch2 > ch1`, `ch1 <> ch2`, etc. To find out what the ASCII characters are compile and run the following program.

```

program ASCIIChars ;
const
    NumChars = 128;
var
    Ch  :  CHAR;
    i   :  INTEGER;
begin
    Writeln('This program prints out the ASCII characters');
    for i := 0 to NumChars - 1 do begin
        Writeln('Char. No. : ', i, ' Char : ', Chr(i));
    end; { FOR }
end.

```

STRING Type Although this is not a scalar type we discuss it now because it is closely related to the CHAR type. A STRING type is an array of characters, i.e., a set of boxes each of which contains a character value. The number of boxes in a Turbo Pascal string is 255.

The STRING type has two operations : $Length(s)$ and the + or *concatenation* operation. For example if `s1`, `s2` and `s3` are variables of type STRING then the program below

```

program Strings;
var s1, s2, s3 :  STRING;
begin
    s1 := 'Tom';
    s2 := 'Jones';
    s3 := s1 + s2;
    Writeln(Length(s1), Length(s2), Length(s3));
    Writeln(s3);
end.

```

would print out the result :

```

3 5 8
TomJones

```

We may mix characters and strings in the same string expression, because characters are regarded as strings of length 1. For example

```
s3 := s1 + ' ' + s2
```

would give the result `Tom Jones`.

We may compare strings with the relational operators. For example, the test `s1 < s2` would give the result `false` because `Tom` is greater than `Jones` in alphabetical order.

3.2 Data Declarations

Data declarations allow the programmer to name (put identifiers on) boxes in memory and to specify the type of data these will hold. There are two broad categories of data : (i) constants and (ii) variables.

Constants are named boxes in memory whose values are set in a `CONST` data declaration and whose values cannot be changed once they are declared.

Variables are also named boxes in memory whose values are set and can be changed as a program is running.

3.2.1 Constant Declarations

The syntax of a constant declaration is as follows :

```
const
  <identifier> = <value>;
  <identifier> = <value>;
  .
  .
  .
  <identifier> = <value>;
```

EXAMPLE : *Constant Declaration*

```
const
  Pi      = 3.14159;      { a REAL constant }
  Answer = 'Y';         { a CHAR constant }
  NDim    = 200;         { an INTEGER constant}
  Test    = true;        { a BOOLEAN constant}
  Exam3   = 'Third Commerce'; { a STRING constant}
```

□

Notice that the *type* of the constant is determined by the type of the value on the right-hand side of the = sign.

3.2.2 Variable Declarations

The syntax of a variable declaration is as follows :

```
var
  <identifier> : <type>;
  <identifier> : <type>;
  .
  .
  .
  <identifier> : <type>;
```

EXAMPLE : *Variable Declaration*

```
var
  x      : REAL;
  y      : REAL;
  i      : INTEGER;
  Name   : STRING;
  Age    : INTEGER;
  Over40 : BOOLEAN;
  Reply  : CHAR;
```

□

Notice that no value is given to a variable in these declarations – only the type is specified. The variables obtain their values through *assignment* or *input* statements which occur in the statement part of a program. These are covered in the next chapter.

Although we have not discussed enough elements of Pascal to understand a complete program let us write a small complete program to show how constant and variable declarations are used.

EXAMPLE : *A Complete Program*

```
program ConstsAndVars;
const
  Pi      = 3.14159;           { a REAL constant }
  Answer = 'Y';               { a CHAR constant }
  NDim    = 200;              { an INTEGER constant}
  Test    = true;             { a BOOLEAN constant}
  Exam3   = 'Third Commerce'; { a STRING constant}
var
  x       : REAL;
  y       : REAL;
  i       : INTEGER;
  Name    : STRING;
  Age     : INTEGER;
  Over40  : BOOLEAN;
  Reply   : CHAR;
begin {This is the statement part of the program}

  Write('Do you wish to continue?');
  Readln(Reply);
  if (Reply = 'Y') then begin
    Write('Type in two real values :');
    Readln(x, y);

    Test := (x < y);
    if Test then begin
      Writeln('x is less than y');
    end
    else begin
      Writeln(' x is greater than y');
    end {IF}

    Writeln('The value of Pi = ', Pi);

    y := 2.0*Pi*x; { The literal constant 2.0 has no name –
                   it is anonymous}

    Writeln(y);
  end; {IF}
end.
```

□

Notice that we have declared some constants and variables that have not been used in the statement part of the program. This is legal in Pascal but not a good idea : declare only those things that you are going to use. Unused constants and variable declarations only clutter up the program and make it difficult to read.

As an aid to readability notice that we have used **boldface** for reserved words and have *indented* some of the statements.

Laboratory Exercise No. 3: ASCII Codes.

This exercise comprises two small programs :

1. Modify PROGRAM ASCIIChars above so that it prints out the *alphabetic* characters, which begin at ASCII code number 65.
2. Write a Pascal program that accepts 3 REAL numbers, computes their average, and writes out the result.

NOTES : Always put the following statements anywhere you wish to pause the output to the screen for viewing.

```
Write('Press RETURN to continue...');  
Readln;
```
