Variables declared inside a routine are only visible within the routine – and to nested routines.

Declarations using the var keyword in the interface section of a unit are visible within the unit and wherever the unit is present in a uses clause.

Declarations using the var keyword in the implementation section of a unit are visible within the unit.

Objects implementing interfaces are reference counted. They are destroyed when their reference count reaches zero. All other objects and any allocated memory must be explicitly destroyed/released after use.

### Variable Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>1</td>
<td>false or true</td>
</tr>
<tr>
<td>Byte</td>
<td>1</td>
<td>0.255</td>
</tr>
<tr>
<td>Cardinal</td>
<td>4</td>
<td>0.4294967295</td>
</tr>
<tr>
<td>Char</td>
<td>1</td>
<td>Extended ASCII</td>
</tr>
<tr>
<td>Currency</td>
<td>8</td>
<td>±4.22E14</td>
</tr>
<tr>
<td>Double</td>
<td>8</td>
<td>5E-324..1.7E308</td>
</tr>
<tr>
<td>Extended</td>
<td>10</td>
<td>3.6E-4951..1.1E4932</td>
</tr>
<tr>
<td>Integer</td>
<td>4</td>
<td>-1247483648..2147483647</td>
</tr>
<tr>
<td>Int64</td>
<td>8</td>
<td>-2^63..2^63 - 1</td>
</tr>
</tbody>
</table>

- Integer
- Long
- Short
- Integer

### Special Constants

- false, true, nil, MAXWORD, MAXINT, MAXDOUBLE, etc.

### Enumerations

e.g. type TDelphiVersion = (dv5 = 5,dv6,dv7,dv8)

Enumerations can be manipulated using inc, dec, pred and succ. ord can be used to get their ordinal value. Prepend enumeration members with two or more lowercase characters identifying their parent enumeration.

Enumerated values require one or more depending on the number of members in the parent enumeration.

### Arrary Types

Any ordinal type can be used to define an array type. e.g.

- VTVersions = array[TDelphiVersions] of String;
- TLevels = array[3..3] of Integer;
- TLetters = array[‘a’..’z’] of Char;
- TInfo = array[Boolean,0..9] of PChar;

### Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>3 + 2</td>
<td>5</td>
</tr>
<tr>
<td>-</td>
<td>3 - 2</td>
<td>-1</td>
</tr>
<tr>
<td>*</td>
<td>3 * 2</td>
<td>6</td>
</tr>
<tr>
<td>/</td>
<td>3 / 2</td>
<td>1.5</td>
</tr>
<tr>
<td>div</td>
<td>3 div 2</td>
<td>1</td>
</tr>
<tr>
<td>mod</td>
<td>3 mod 2</td>
<td>3 (3 div 2)^2</td>
</tr>
</tbody>
</table>

### Conversion from Strings

- StrToDef(s,def) – s to currency. def on error.
- StrToInt(s,def) – s to integer. def on error.
- StrToDateTimeDef(s,def) – s to real. def on error.
- StringToStr(Documented) – returns a datetime object. str in hex in time fraction of N digits

### Date & Time Routines

- Date – current date, time fraction set to zero.

- DecodeDate(Year,M,D) – year, month & day to YMD
- EncodeDateTime(Year,Month,Day) – hrs, mins, s & ms to HMSN
- FormatDateTime(ptrn,datetime) – returns datetime as string formatted using ptrn.
- EncodeHex(value,N) – value in hexadecimal with N digits

- FormatDateTime(ptrn,datetime) – returns formatted date string.
Drive/Folder/Path Manipulation

Drive.CreateDir(path) – creates directory. false on error.
Drive.DirectoryExists(path) – true if dir exists.
Drive.ExtractFilePath(AFile) – returns everything before filename.ext.
Drive.ExtractFileExt(AFile) – returns filename.ext.
Drive.FreeDirectories(path) – creates all directories in path. false on error.
Drive.GetDir() – current directory.
Drive.RemoveDir(path) – removes dir.

Execution/Flow Control

SysUtils.socket – raise silent exception.
repeat – break from loop (for, repeat or while).
continue – continue to next iteration of loop.
halt – immediate termination of program.

Number Manipulation

Math.abs(arg) – returns absolute value.
Math.ceil(arg) – lowest integer >= arg.
Math.floor(arg) – highest integer <= arg.
Math.frac(arg) – fractional part of N.
Math.int(N) – integer part of real number N.
Math.log10(N) – log to the base 10 of N.
Math.log2(N) – log to the base 2 of N.
Random.Random – random number in the range 0..1.
Randomize – initialize random number generator.
RandSeed – Seed value for random number generator.
Round(N) – nearest whole number. Midway values rounded to even number.
Math.RoundToN(d) – round N to 10^d.

Ordnal Manipulation

dec(arg,N) – decrements ordinal arg by N.
inc(arg,N) – increments ordinal arg by N.
low(arg) – low bound of arg type.
ord(arg) – ordinal value of boolean, char or enumerated arg.
pred(arg) – predecessor of ordinal type arg.
succ(arg) – subsequent value of ordinal type arg.

String Manipulation

chr(arg) – ASCII character at arg.
SysUtils.CompareStr(s1,s2) – case sensitive comparison. s1 < s2 returns -1, s1 = s2 returns 0 & s1 > s2 returns 1.
SysUtils.CompareToText(s1,s2) – case insensitive comparison. Returns as above.
Copy(s,Index,Count) – Copy characters in s starting at Index.
Delete(s,Index,Count) – deletes Count characters in s starting at Index.
StrUtils.LeftStr(s,Count) – Count characters in s starting from the left. RightStr is similar.
StrUtils.MiddleStr(s,Index,Count) – Count characters in s starting from Index.
Length(s) – number of characters in s.
SysUtils.LowerCase(s) – s in lower case. UpperCase is similar.
SysUtils.TextSize(s) – returns true if s1 = s2, not case sensitive. Returns true or false.
SetLength(s,len) – sets length of string s to len.
StringOfChar(Char,Count) – returns string containing Count Chars.

Variant Manipulation

VarFromDateTime(date) – date as a variant.
VarToDateTime(V) – V as TDateTime.
VarAsString(V,AType) – V converted to variant of type AType.
VarToStr(V) – V as a string.
VarToWideStr(V) – V as a wide string.
VarType(V) – variant type of V.

Format Specifiers

Date/Time Formats

• c – ShortDate/Time

• d – day, no leading zero.
• dd – day, leading zero if necessary.
• ddd – Short day names.
• dddd – Long day names.
• m, mm, mmm, mmmm – Month names, as above.
• yy – two digit year.
• yyyy – four digit year.
• h, n, s – hour, minute & second. No leading zero.
• hh, nn, ss – hour, minute & second with leading zero
• tt – ShortTimeFormat
• tt – LongTimeFormat
• am/pm – Use 12h clock. Follow hhh by am or pm.
• ampm – Use 12h clock. Follow hhh by TimeAM/PMString global variables.
• | – date separator as in DateSeparator global variable
• : – time separator as in TimeSeparator global variable.
• “xx” or “xx” – literal characters

Format function specifiers

Format strings consist of one or more specifiers bearing the form [%[%%][[w]d].dL]

• i – indicates left justification. (The default is right)
• *i indicates the total character width of the output value.
• w indicates the total character width of the output value. If necessary this is padded out with spaces – right or left depending on the justification specifier.
• d is the precision specifier. The meaning of this depends on the nature of the quantity being formatted.

• The number of characters in integers & hexadecimal integers.
• The number of decimals in real numbers in general, f, format.
• The number of decimals + the E in real numbers in scientific format.
• The number of characters in a string.

• l indicates that nature. d for integer, f for real, e for scientific, n real but with thousands separators, s for string and x or “xx” – literal characters

Example

%ld Simple Integer formatting
%0.nd Integer with n digits – padded if shorter
%m.n Integer with n digits in a width of m. m is ignored if insufficient.
%m.nf Floating point number, width m with n decimal digits.
%-m.nf As above but left justified.
%m.ns String formatted to a width of m characters and containing n characters. Truncated if n is less than string length. n is ignored if greater than string length.
%sm.nx Integer in hexadecimal format. Rest as for %ld, above.

Other options exist.

Conditional Execution/Bracing

if Condition then
begin
Code
end;
else
begin
Code
end;

case selector of
begin
caseList1:code;
caseList2:code;
...
caseListN:code;
[else code];
end;

Looping

for i:=LowBound to HighBound do
begin
Code;
end;

for i:=HighBound downto LowBound do
begin
Code;
end;

break
Code;
until Condition;
while Condition
begin
Code;
end;

Dispense with the begin & end to execute a single line of code. repeat loops execute at least once. Use break, continue or exit to modify/terminate loop execution.

Notes

• MAXDOUBLE etc are defined in Math.

• Unless preceded by Unit, the routine is in SysUtils.

• Unless preceded by Unit, the routine is in System.

• For widestrings use the same function but preceded by Wide, e.g. WideString.

Color Codes

blue – Delphi keyword.
green – Delphi routine (function or procedure)

Math. – unit to be specified in uses clause. Does not apply to System.

An extensive range of free quick reference cards is available at http://www.explainth.at