Delphi Technical Reference Card VII

GulfCoastal.bizland.com

Symbolic constant name	*Value (hexadecimal)*	*Mouse/keyboard equivalent*
VK_LBUTTON	01	Left mouse button
VK_RBUTTON	02	Right mouse button
VK_CANCEL	03	Control-break processing
VK_MBUTTON	04	Middle mouse button
	05–07	Undefined
VK_BACK	08	BACKSPACE key
VK_TAB	09	TAB key
	0A–0B	Undefined
VK_CLEAR	0C	CLEAR key
VK_RETURN	0D	ENTER key
	0E–0F	Undefined
VK_SHIFT	10	SHIFT key
VK_CONTROL	11	CTRL key
VK_MENU	12	ALT key
VK_PAUSE	13	PAUSE key
VK_CAPITAL	14	CAPS LOCK key
	15–19	Resv Kanji systems
	1A	Undefined
VK_ESCAPE	1B	ESC key
	1C–1F	Resv Kanji systems
VK_SPACE	20	SPACEBAR
VK_PRIOR	21	PAGE UP key
VK_NEXT	22	PAGE DOWN key
VK_END	23	END key
VK_HOME	24	HOME key
VK_LEFT	25	LEFT ARROW key
VK_UP	26	UP ARROW key
VK_RIGHT	27	RIGHT ARROW key
VK_DOWN	28	DOWN ARROW key
VK_SELECT	29	SELECT key
	2A	OEM specific
VK_EXECUTE	2B	EXECUTE key
VK_SNAPSHOT	2C	Print Screen key
VK_INSERT	2D	INS key
VK_DELETE	2E	DEL key
VK_HELP	2F	HELP key
VK_0	30	0 key
VK_1	31	1 key
VK_2	32	2 key
VK_3	33	3 key
VK_4	34	4 key
VK_5	35	5 key
VK_6	36	6 key
VK_7	37	7 key
VK_8	38	8 key
VK_9	39	9 key
	3A–5A	Undefined
VK_LWIN	5B	Left Windows key (MS Keybd)
VK_RWIN	5C	Right Windows key (MS Keybd)
VK_APPS	5D	Applications key (MS Keybd)
	5E–5F	Undefined
VK_NUMPAD0	60	Numeric keypad 0 key
VK_NUMPAD1	61	Numeric keypad 1 key
VK_NUMPAD2	62	Numeric keypad 2 key
VK_NUMPAD3	63	Numeric keypad 3 key
VK_NUMPAD4	64	Numeric keypad 4 key
VK_NUMPAD5	65	Numeric keypad 5 key
VK_NUMPAD6	66	Numeric keypad 6 key
VK_NUMPAD7	67	Numeric keypad 7 key
VK_NUMPAD8	68	Numeric keypad 8 key
VK_NUMPAD9	69	Numeric keypad 9 key
VK_MULTIPLY	6A	Multiply key
VK_ADD	6B	Add key
VK_SEPARATOR	6C	Separator key
VK_SUBTRACT	6D	Subtract key
VK_DECIMAL	6E	Decimal key
VK_DIVIDE	6F	Divide key
VK_F1	70	F1 key
VK_F2	71	F2 key
VK_F3	72	F3 key
VK_F4	73	F4 key
VK_F5	74	F5 key
VK_F6	75	F6 key
VK_F7	76	F7 key
VK_F8	77	F8 key
VK_F9	78	F9 key
VK_F10	79	F10 key
VK_F11	7A	F11 key
VK_F12	7B	F12 key
VK_F13	7C	F13 key
VK_F14	7D	F14 key
VK_F15	7E	F15 key
VK_F16	7F	F16 key
VK_F17	80H	F17 key
VK_F18	81H	F18 key
VK_F19	82H	F19 key
VK_F20	83H	F20 key
VK_F21	84H	F21 key
VK_F22	85H	F22 key
VK_F23	86H	F23 key
VK_F24	87H	F24 key
	88–8F	Unassigned
VK_NUMLOCK	90	NUM LOCK key
VK_SCROLL	91	SCROLL LOCK key
	92–B9	Unassigned
	BA–C0	OEM specific
	C1–DA	Unassigned
	DB–E4	OEM specific
	E5	Unassigned
	E6	OEM specific
	E7–E8	Unassigned
	E9–F5	OEM specific
VK_ATTN	F6	Attn key
VK_CRSEL	F7	CrSel key
VK_EXSEL	F8	ExSel key
VK_EREOF	F9	Erase EOF key
VK_PLAY	FA	Play key
VK_ZOOM	FB	Zoom key
VK_NONAME	FC	Reserved for future use.
VK_PA1	FD	PA1 key
VK_OEM_CLEAR	FE	Clear key
	FF	Unassigned

File:

```
FPath := GetCurrentDir;
OpenDialog1.InitialDir := FPath;

ExtractFileDrive('<file name>')      C:
ExtractFileDir('<file name>')        C:\<path>
ExtractFilePath('<file name>')       C:\<path>\
ExtractFileName('<file name>')       fname.ext
ExtractFileExt('<file name>')        .ext
ExtractFilePath(Application.ExeName) app path

DirectoryExists('<folder>')    [use FileCtrl]
CreateDir('<folder>')

OpenDialog1.Filter :=
   'Text Files (*.txt)|*.txt|All (*.*)|*.*';
OpenDialog1.FilterIndex := 1; List .txt files
OpenDialog1.Execute;
```

Format Strings:

```
Format('%.3d', [<integer: 4>]);       '004'
Format('%2.2d%2.2d%4d', [1,1,2000]); '01012000'
Format('%.0n', [<real: 1234567>]);   '1,234,567'
Format('%.2n', [<real: 12345.675>]); '12,345.68'
Format('%m', [<real: 12.34567>]);    '$12.35'
Format('%x', [<integer: 43>]);       '2B'
Format('%p', [<pointer>]);           '8 chr adr'
Format('%s string.', ['Some']);   'Some string.'
Format('{%-4.3s} {%4.2s}', ['L123', 'R123']);
                                   '{L12 } {  R1}'
Format('%2:s %1:s %0:s', ['1st', '2nd', '3rd']);
                                    '3rd 2nd 1st'
Format('{%*.*f}', [<len: 9>, <dec: 4>, 100*PI]);
                                    '{ 314.1593}'
FloatToStrF(123.45, ffFixed, <len: 4>,<dec: 1>);
                                          '123.5'
FormatMaskText('0-00-00;0;_', '12345');'1-23-45'
FormatFloat('#00,000.0##', 1234.400); '01,234.4'
```

Date/Time Formats:

```
 FormatDateTime('mm/dd/yyyy', Now); '09/07/2000'
 FormatDateTime('hh:n:ss', Now);    '09:5:59'
 FormatDateTime('<Specifier>', Now);
 <c> 7/29/00 5:24:08 PM;
 <m> 7; <mm> 07; <mmm> Jul; <mmmm> July;
 <d> 1; <dd> 01; <ddd> Sun; <dddd> Sunday;
 <ddddd> 7/9/00; <dddddd> Sunday, July 09, 2000;
 <yy> 00; <yyyy> 2000;
 <h> 9; <hh> 09; <n> 7; <nn> 07; <s> 9; <ss> 09;
 <t> 5:38 PM; <tt> 5:38:28 PM;
 <am/pm> pm; <a/p> a; <ampm> PM; </> /; <:> :
```

String Manipulation:

```
 Chr(<Integer>);
 Copy(<SourceString>, <start pos>, <length>);
 CompareStr(<SourceString1>, <SourceString2>);
 Delete(<SourceString>, <start pos>, <length>);
 IntToStr(<SourceInteger>);
 Insert(<fromSourceString>, <toSourceString>,
        <start pos>);
 Length(<SourceString>);
 Pos('<find this>', <SourceString>);
 SetLength(<SourceString>, <length>);
 StringOfChar('<Character>', <quantity>);
 StrPas(<PCharString>);
 StrPCopy(<SourceString>);
 StrToInt(<SourceString>);
 StrToIntDef(<SourceString>, <DefaultInteger>);
 StrTo<???>(<SourceString>);
 <???>ToStr(<Source???>); ??? = Float, Currency,
                           Date, Time, DateTime
 StringReplace(<SourceString>, '<replace this>',
              '<with this>', [rfReplaceAll]);
 Trim(<SourceString>);      trim l/r blanks
 TrimLeft(<SourceString>);  trim left blanks
 TrimRight(<SourceString>); trim right blanks
 LowerCase(<SourceString>);
 UpperCase(<SourceString>);
 UpCase(<Char>);
```

Sets:

```
 ThisSet : set of byte;    [0-255]
 ThisSet := [1, 2, 3, 7];  initialize to 1,2,3,7
 ThisSet := ThisSet - [3]; exclude number 3
 ThisSet := ThisSet + [5]; include number 5
 ThisSet := [];            purge all numbers
 if 7 in ThisSet ...
```

Pointer:

```
 Pt : pointer; CharSet := 'AbCd'; Data : string;
 Pt := @CharSet;
 Data := PChar(Pt^);      Data = 'AbCd'
 Data := PChar(Pt^)[0];   Data = 'A'
```

Math Expressions:

```
 Absolute value:      x := Abs(x);
 Addition:            x := y + z;
 Address of operator: ptr := @ThisRecord;
 Array subscript operator: x := ThisArray[5];
 Assignment:          x := 10;
 Bitwise AND:         x := x AND $02;
 Bitwise NOT:         x := x AND NOT $02;
 Bitwise OR:          x := x OR $FF;
 Bitwise SHL:         x := x SHL $02;
 Bitwise SHR:         x := x SHR $02;
 Bitwise XOR:         x := x XOR y;
 Decrement:           Dec(x);  Dec(x, 2);
 Equal to:            if (x = 10) ...
 Fraction return:     x := Frac(x);
 Greater than or equal to: if (x >= 10) ...
 Greater than:        if (x > 10) ...
 Hex value operator:  x := $FF;
 Increment:           Inc(x);  Inc(x, 2);
 Integer division:    x := y Div 10;
 Less than or equal to: if (x <= 10) ...
 Less than:           if (x < 10) ...
 Logical AND:         if (x = 1) And (y = 2)..
 Logical NOT:         if Not Valid then ...
 Logical OR:          if (x = 1) Or (y = 2)...
 Maximum number return:     x := Max(x, y);
 Membership (dot) operator: x := Record.Data;
 Minimum number return:     x := Min(x, y);
 Multiplication:      x := y * z;
 Not equal to:        if (x <> 10) ...
 Odd number:          if Odd(9) ...
 Ord:                 x := Ord('<character>');
 Pi:                  x := Pi;
 Pointer operator:    ThisObject.Data^;
 Real division:       x := y / 3.14;
 Remainder:           x := y Mod 2;
 Round to negative:   x := Floor(x);
 Round to positive:   x := Ceil(x);
 Square:              x := Sqr(x);
 Square root:         x := Sqrt(x);
 Subtraction:         x := y - z;
 Return integer rounded toward zero:
                    FloatValue := Int(Real);
 Discard decimals and return Integer:
                    Int64Value := Trunc(Real);
 Round to the nearest whole number:
                    Int64Value := Round(Real);
```

Numeric Variables:

Type	Size	Range of Values
Boolean	1	True or False
Byte	1	0 to 255
Cardinal	4	0 to 4,294,967,295
Char	1	0 to 255
Comp	8	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Currency	8	–922,337,203,685,477.5808 to 922,337,203,685,477.5807
Double	8	5.0 ¥ 10^{-324} to 1.7 ¥ 10^{308}
Extended	10	3.4 ¥ 10^{-4932} to 1.1 ¥ 10^{4932}
Int64	8	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Integer	4	–2,147,483,648 to 2,147,483,647
LongInt	4	–2,147,483,648 to 2,147,483,647
LongWord	4	0 to 4,294,967,295
Real	8	5.0 ¥ 10^{-324} to 1.7 ¥ 10^{308}
ShortInt	1	–128 to 127
Single	4	1.5 ¥ 10^{-45} to 3.4 ¥ 10^{38}
SmallInt	2	–32,768 to 32,767
WideChar	2	0 to 65,535
Word	2	0 to 65,535
Variant	16	All above

Delphi Technical Reference Card VII
GulfCoastal.hypermart.net

#	$						
Dec	*Hex*	*Fn*	*Binary*	*Dec*	*Hex*	*Fn*	*Binary*
00	00		0000 0000	64	40	@	0100 0000
01	01		0000 0001	65	41	A	0100 0001
02	02		0000 0010	66	42	B	0100 0010
03	03		0000 0011	67	43	C	0100 0011
04	04		0000 0100	68	44	D	0100 0100
05	05		0000 0101	69	45	E	0100 0101
06	06		0000 0110	70	46	F	0100 0110
07	07	BE	0000 0111	71	47	G	0100 0111
08	08	BK	0000 1000	72	48	H	0100 1000
09	09	TAB	0000 1001	73	49	I	0100 1001
10	0A	LF	0000 1010	74	4A	J	0100 1010
11	0B	VT	0000 1011	75	4B	K	0100 1011
12	0C	FF	0000 1100	76	4C	L	0100 1100
13	0D	CR	0000 1101	77	4D	M	0100 1101
14	0E		0000 1110	78	4E	N	0100 1110
15	0F		0000 1111	79	4F	O	0100 1111
16	10		0001 0000	80	50	P	0101 0000
17	11		0001 0001	81	51	Q	0101 0001
18	12		0001 0010	82	52	R	0101 0010
19	13		0001 0011	83	53	S	0101 0011
20	14		0001 0100	84	54	T	0101 0100
21	15		0001 0101	85	55	U	0101 0101
22	16		0001 0110	86	56	V	0101 0110
23	17		0001 0111	87	57	W	0101 0111
24	18		0001 1000	88	58	X	0101 1000
25	19		0001 1001	89	59	Y	0101 1001
26	1A	EOF	0001 1010	90	5A	Z	0101 1010
27	1B	ESC	0001 1011	91	5B	[	0101 1011
28	1C		0001 1100	92	5C	\	0101 1100
29	1D		0001 1101	93	5D	]	0101 1101
30	1E		0001 1110	94	5E	^	0101 1110
31	1F		0001 1111	95	5F	_	0101 1111
32	20	SP	0010 0000	96	60	`	0110 0000
33	21	!	0010 0001	97	61	a	0110 0001
34	22	"	0010 0010	98	62	b	0110 0010
35	23	#	0010 0011	99	63	c	0110 0011
36	24	$	0010 0100	100	64	d	0110 0100
37	25	%	0010 0101	101	65	e	0110 0101
38	26	&	0010 0110	102	66	f	0110 0110
39	27	'	0010 0111	103	67	g	0110 0111
40	28	(	0010 1000	104	68	h	0110 1000
41	29	)	0010 1001	105	69	i	0110 1001
42	2A	*	0010 1010	106	6A	j	0110 1010
43	2B	+	0010 1011	107	6B	k	0110 1011
44	2C	,	0010 1100	108	6C	l	0110 1100
45	2D	-	0010 1101	109	6D	m	0110 1101
46	2E	.	0010 1110	110	6E	n	0110 1110
47	2F	/	0010 1111	111	6F	o	0110 1111
48	30	0	0011 0000	112	70	p	0111 0000
49	31	1	0011 0001	113	71	q	0111 0001
50	32	2	0011 0010	114	72	r	0111 0010
51	33	3	0011 0011	115	73	s	0111 0011
52	34	4	0011 0100	116	74	t	0111 0100
53	35	5	0011 0101	117	75	u	0111 0101
54	36	6	0011 0110	118	76	v	0111 0110
55	37	7	0011 0111	119	77	w	0111 0111
56	38	8	0011 1000	120	78	x	0111 1000
57	39	9	0011 1001	121	79	y	0111 1001
58	3A	:	0011 1010	122	7A	z	0111 1010
59	3B	;	0011 1011	123	7B	{	0111 1011
60	3C	<	0011 1100	124	7C	|	0111 1100
61	3D	=	0011 1101	125	7D	}	0111 1101
62	3E	>	0011 1110	126	7E	~	0111 1110
63	3F	?	0011 1111	127	7F		0111 1111

(Fn:Courier) (BE:Bell BK:BackSpace TAB:Tab LF:LineFeed VT:VerticalTab FF:FormFeed CR:CarriageReturn EOF:EndOfFile ESC:Escape SP:space [#13#10:LineBreak])

Dec	*Hex*	*Fn*	*Binary*	*Dec*	*Hex*	*Fn*	*Binary*
128	80		1000 0000	192	C0	À	1100 0000
129	81		1000 0001	193	C1	Á	1100 0001
130	82		1000 0010	194	C2	Â	1100 0010
131	83		1000 0011	195	C3	Ã	1100 0011
132	84		1000 0100	196	C4	Ä	1100 0100
133	85		1000 0101	197	C5	Å	1100 0101
134	86		1000 0110	198	C6	Æ	1100 0110
135	87		1000 0111	199	C7	Ç	1100 0111
136	88		1000 1000	200	C8	È	1100 1000
137	89		1000 1001	201	C9	É	1100 1001
138	8A		1000 1010	202	CA	Ê	1100 1010
139	8B		1000 1011	203	CB	Ë	1100 1011
140	8C		1000 1100	204	CC	Ì	1100 1100
141	8D		1000 1101	205	CD	Í	1100 1101
142	8E		1000 1110	206	CE	Î	1100 1110
143	8F		1000 1111	207	CF	Ï	1100 1111
144	90		1001 0000	208	D0	Ð	1101 0000
145	91		1001 0001	209	D1	Ñ	1101 0001
146	92		1001 0010	210	D2	Ò	1101 0010
147	93		1001 0011	211	D3	Ó	1101 0011
148	94		1001 0100	212	D4	Ô	1101 0100
149	95		1001 0101	213	D5	Õ	1101 0101
150	96		1001 0110	214	D6	Ö	1101 0110
151	97		1001 0111	215	D7	×	1101 0111
152	98		1001 1000	216	D8	Ø	1101 1000
153	99		1001 1001	217	D9	Ù	1101 1001
154	9A		1001 1010	218	DA	Ú	1101 1010
155	9B		1001 1011	219	DB	Û	1101 1011
156	9C		1001 1100	220	DC	Ü	1101 1100
157	9D		1001 1101	221	DD	Ý	1101 1101
158	9E		1001 1110	222	DE	Þ	1101 1110
159	9F		1001 1111	223	DF	ß	1101 1111
160	A0		1010 0000	224	E0	à	1110 0000
161	A1	¡	1010 0001	225	E1	á	1110 0001
162	A2	¢	1010 0010	226	E2	â	1110 0010
163	A3	£	1010 0011	227	E3	ã	1110 0011
164	A4	¤	1010 0100	228	E4	ä	1110 0100
165	A5	¥	1010 0101	229	E5	å	1110 0101
166	A6	¦	1010 0110	230	E6	æ	1110 0110
167	A7	§	1010 0111	231	E7	ç	1110 0111
168	A8	¨	1010 1000	232	E8	è	1110 1000
169	A9	©	1010 1001	233	E9	é	1110 1001
170	AA	ª	1010 1010	234	EA	ê	1110 1010
171	AB	«	1010 1011	235	EB	ë	1110 1011
172	AC	¬	1010 1100	236	EC	ì	1110 1100
173	AD		1010 1101	237	ED	í	1110 1101
174	AE	®	1010 1110	238	EE	î	1110 1110
175	AF	¯	1010 1111	239	EF	ï	1110 1111
176	B0	°	1011 0000	240	F0	ð	1111 0000
177	B1	±	1011 0001	241	F1	ñ	1111 0001
178	B2	²	1011 0010	242	F2	ò	1111 0010
179	B3	³	1011 0011	243	F3	ó	1111 0011
180	B4	´	1011 0100	244	F4	ô	1111 0100
181	B5	µ	1011 0101	245	F5	õ	1111 0101
182	B6	¶	1011 0110	246	F6	ö	1111 0110
183	B7	·	1011 0111	247	F7	÷	1111 0111
184	B8	¸	1011 1000	248	F8	ø	1111 1000
185	B9	¹	1011 1001	249	F9	ù	1111 1001
186	BA	º	1011 1010	250	FA	ú	1111 1010
187	BB	»	1011 1011	251	FB	û	1111 1011
188	BC	¼	1011 1100	252	FC	ü	1111 1100
189	BD	½	1011 1101	253	FD	ý	1111 1101
190	BE		1011 1110	254	FE	þ	1111 1110
191	BF	¿	1011 1111	255	FF	ÿ	1111 1111

(Fn:Courier)

v7.20

Bitwise Operations:

```
     [Byte: 1111 0000  Mask Order: 7654 3210]
 Byte OR (1 SHL 0);           set  bit 1 (1111 0001)
 Byte AND (NOT (1 SHL 5)); zero bit 6 (1101 0000)
 Byte XOR (1 SHL 7);     toggle bit 8 (0111 0000)
 IF Bite AND (1 SHL 4) <> 0 ... test if bit 5 set

     [Byte: 1111 0000  Mask Order: 8421 8421]
 Byte OR $01;                 set  bit 1 (1111 0001)
 Byte AND (NOT $20);          zero bit 6 (1101 0000)
 Byte XOR $80;              toggle bit 8 (0111 0000)
 IF Bite AND $10 <> 0 ...       test if bit 5 set
```

For/While/Case Instructions:

```
 for I := 0 to 9 do begin...
 for I := 9 downto 1 do begin...
 while I < 100 do begin...
 case ANumber of 1 : do this...; 2 : begin...end;
 case AString of 'a', 'c'..'z' : do this...;
```

Messages:

```
 MessageBox(0, '<text>',pchar('<title>'),<mb+mb>)
 mbButton: mb_OK  mb_OKCancel mb_AbortRetryIgnore
         mb_YesNo  mb_RetryCancel  mb_YesNoCancel
 mbBitmap: mb_IconExclamation  mb_IconQuestion
           mb_IconInformation  mb_IconError
 mbDefaultButton: mb_DefButton1..mb_DefButton4
 mbModality: mb_ApplModal  mb_SystemModal
             mb_TaskModal
 mbSpecial:  mb_Default_DeskTop_Only  mb_TopMost
             mb_Right mb_SetForeground mb_Help
 idReturnValues: idOK  idCancel  idYes  idNo
                 idAbort  idIgnore  idRetry

 MessageDlg('<text>', <mt>, [<mb>, <mb>], 0)
 mtBitmap: mtWarning  mtError  mtInformation
           mtConfirmation  mtCustom
 mbButtonText: mbOK  mbCancel  mbYes  mbNo  mbAll
               mbAbort  mbRetry  mbIgnore  mbHelp
 mrReturnValues: mrNone  mrOk  mrCancel  mrRetry
            mrYes  mrNo  mrIgnore  mrAbort  mrAll

 InputBox('<caption>', '<text>', '<default str>')

 ShowMessage('<text>', + #13 + '<text>')
```

Non-Standard Colors:

BlueGreen	=	$CCCC00	LtPurple	=	$FBA29D
Brick	=	$003399	LtPurple	=	$FFCCCC
Brown	=	$006699	LtViolet	=	$FFCCFF
Brown	=	$6058A0	MediumGray	=	$A4A0A0
BurntSienna	=	$000088	MoneyGreen	=	$C0DCC0
Butterfly	=	$EF10B8	Mustard	=	$00C4C4
Cosmo	=	$C802F2	NavalBlue	=	$CC9933
Cream	=	$F0FBFF	OliveGreen	=	$009966
DkBlue	=	$770000	Orange	=	$33CCFF
DkGreen	=	$005500	PaleBlue	=	$FFFFCC
DkOrange	=	$0099CC	PaleGreen	=	$79FF91
DkPurple	=	$AE0D3E	PaleYellow	=	$CCFFFF
DkRose	=	$9966FF	PaleYellow	=	$E2FCFB
DkTeal	=	$999933	Peach	=	$647EF9
DkViolet	=	$993399	Pink	=	$8640FB
Editor	=	$950416	Pumpkin	=	$0099FF
Grape	=	$B16778	Purple	=	$CC0099
Green	=	$2BCA56	RedBaron	=	$0033FF
Honey	=	$1CAEE6	Rose	=	$5E24F4
Khaki	=	$669999	Sea	=	$B90F0B
LtBlue	=	$FFCC99	SeaGreen	=	$CCFF00
LtBrown	=	$688FB0	Sky	=	$FD8A4D
LtCyan	=	$FFFF99	SkyBlue	=	$F0CAA6
LtGrape	=	$BD85C7	Slab	=	$B3B67E
LtGreen	=	$CCFFCC	Violet	=	$FF33FF
LtOrange	=	$99CCFF	YellowGreen	=	$00FFCC

Definitions:

Class: a collection of procedures, functions and other fields that make up a specific programming task.

Component: a binary function that performs a predefined function (edit control, list box, etc.).

Event Handler: code invoked as a result of an event.

Events: occur when a user interacts with a component (OnEnter, OnExit, etc.).

Function: a section of code that performs some task and returns a value.

Method: a Procedure or Function that is a member of a class.

Object: a binary portion of a program that performs a specific programming task.

Object Pascal: Borland modified Pascal language that extended Pascal, creating a new language.

Parameter: a value passed to a Procedure or Function.

Pointer: a variable that holds the address of another variable.

Procedure: a section of code that performs some task but does not returns a value.

Properties: control how a component operates (color, width, etc.).

Unit: a text file of Delphi code that is compiled into machine code.

Uses List: a list of external units referenced by a Unit.

Compiler Directives:

```
 {$DEFINE $name} set to True
 {$UNDEF  $name} set to False
 {$IF $name} ... {$ELSE} ... {$ENDIF}

 {$DEFINE anyname}
 {$IFDEF  anyname} ... {$ENDIF}
```

Standard Conditional Symbols:

```
 {$DEFINE Debug}    {$IFDEF Debug}
 {$DEFINE WIN32}    {$IFDEF WIN32}
 {$DEFINE VER120}   {$IFDEF VER120}
 {$DEFINE CPU386}   {$IFDEF CPU386}
 {$DEFINE CONSOLE}  {$IFDEF CONSOLE}

 (¹default OFF)         {$MAXSTACKSIZE num}
 {$ALIGN}               {$MINENUMSIZE 1}
 {$APPTYPE}             {$MINSTACKSIZE num}
 {$ASSERTIONS}          {$OPENSTRINGS}
 {$BOOLEVAL}¹           {$OPTIMIZATION}
 {$DEBUGINFO}           {$OVERFLOWCHECKS}¹
 {$DEFINITIINFO}        {$R filename.RES}
 {$DESCRIPTION '..'}    {$RANGECHECKS}¹
 {$E extension}         {$REALCOMPATIBILITY}¹
 {$EXTENDEDSYNTAX}      {$SAFEDIVIDE}¹
 {$HINTS}               {$STACKFRAMES}¹
 {$IMAGEBASE number}    {$TYPEDADDRESS}¹
 {$INCLUDE filename}    {$TYPEINFO}¹
 {$IOCHECKS}            {$VARSTRINGCHECKS}
 {$LINK filename}       {$WARNINGS}
 {$LOCALSYMBOLS}        {$WEAKPACKAGEUNIT}¹
 {$LONGSTRINGS}         {$WRITEABLECONST}
```

User: