ภาคผนวก A

Reserved Words, Standard Identifiers, Operators, Units, Functions, Procedures, and Compiler Directives

Reserved Words and Standard Directives

Reserved words are integral parts of Turbo Pascal. They cannot be redefined and must not be declared as user-defined identifiers.

```
and           exports         mod         shr
asm           file            nil         string
array         for             not         then
begin         function        object      to
case          goto            of          type
const         if              or          unit
constructor   implementation  packed      until
destructor    in              procedure   uses
div           inherited       program     var
do            inline          record      while
downto        interface       repeat      with
else          label           set         xor
end           library         shl
```

The following are Turbo Pascal's standard directives. Unlike reserved words, you may redefine them, but we do not advise this.

```
absolute      far             near        virtual
assembler     forward         private
external      interrupt       public
```

`private` and `public` act as reserved words within object type declarations, but are otherwise treated as directives.

Selected Standard Identifiers

Turbo Pascal defines a number of standard identifiers for predefined types, constants, variables, procedures, and functions. Any standard identifier may be redefined but it will mean loss of the facility offered by that identifier and may lead to confusion.

```
Units
  Crt, Dos, Graph, Overlay, Printer, System

Constants
  False, True, MaxInt, MaxLongInt

Types
  Boolean, Char, Text, Integer, ShortInt, LongInt, Byte,
  Word, Real, Single, Double, Extended, Comp

Files
  Input, Output

Functions
  Abs, ArcTan, Chr, Cos, Concat, Copy, EOF, EOLN, Exp,
  FileSize, Frac, Int, IOResult, Ln, Length, Odd, Ord, Pi, Pos,
  Pred, Random, Round, Sin, Sqr, Sqrt, Succ, Trunc, Upcase

Procedures
  Assign, Close, Delete, Dispose, Erase, Exit, Halt, Insert,
  New, Randomize, Read, ReadLn, Reset, Rewrite, Seek, Str,
  Val, Write, WriteLn
```

Operators

Table B.1 summarizes all the operators of Turbo Pascal. The operators are grouped in order of descending precedence. If the Operand Type and Result Type columns contain `Integer, Real`, the result type is `Real` unless both operands are integers. *Scalar types* are all ordinal and `Real` data types.

Table B.1
Table of Operators

Operator	Operation	Operand Type(s)	Result Type
+ unary	Sign identity	`Integer, Real`	Same as operand
– unary	Sign inversion	`Integer, Real`	Same as operand
@	Operand address	Variable reference or procedure or function identifier	Pointer
`not`	Negation	`Integer, Boolean`	Same as operand
*	Multiplication	`Integer, Real`	`Integer, Real`
	Set intersection	Any set type	Same as operand
/	Division	`Integer, Real`	`Real`
`div`	Integer division	`Integer`	`Integer`
`mod`	Modulus (remainder)	`Integer`	`Integer`
`and`	Arithmetical and	`Integer`	`Integer`
	Logical and	`Boolean`	`Boolean`
`shl`	Shift left	`Integer`	`Integer`
`shr`	Shift right	`Integer`	`Integer`
+	Addition	`Integer, Real`	`Integer, Real`
	Concatenation	`string` or `Char`	`string`
	Set union	Any set type	Same as operand

Table B.1
Table of Operators
(Cont.)

Operator	Operation	Operand Type(s)	Result Type
-	Subtraction	Integer, Real	Integer, Real
	Set difference	Any set type	Same as operand
or	Arithmetical or	Integer	Integer
	Logical or	Boolean	Boolean
xor	Arithmetical xor	Integer	Integer
	Logical xor	Boolean	Boolean
=	Equality	Any scalar type	Boolean
		string	Boolean
		Any set type	Boolean
		Any pointer type	Boolean
<>	Inequality	Any scalar type	Boolean
		string	Boolean
		Any set type	Boolean
		Any pointer type	Boolean
>=	Set inclusion	Any set type	Boolean
	Greater than or equal	Any scalar type	Boolean
		string	Boolean
<=	Set inclusion	Any set type	Boolean
	Less than or equal	Any scalar type	Boolean
		string	Boolean
>	Greater than	Any scalar type	Boolean
		string	Boolean
<	Less than	Any scalar type	Boolean
		string	Boolean
in	Set membership	The first operand may be of any ordinal type, the second must be a set of elements of that type.	Boolean

Units

Turbo Pascal is distributed with several predefined units, similar to those that you might define yourself, containing a large number of additional constants, types, functions, and procedures. Some of these predefined units are described in Table B.2. All but the `Graph` unit are stored in the file `TURBO.TPL`. The details of the contents of each unit are described in the *Borland Pascal Reference Guide* and also in the on-line help facility provided in the Turbo Pascal integrated environment.

Table B.2
Table of Standard Units

Unit	Description
Crt	Contains routines that allow you full control over the PC's screen display, keyboard, and sound
Dos	Supports several DOS functions, including date-and-time control, directory search, and program execution

Table B.2
Table of Standard Units
(Cont.)

Unit	Description
`Graph`	Stored in the file `GRAPH.TPU`, contains a library of 50 graphics routines and device-independent graphics support for several display devices
`Overlay`	Contains the Turbo Pascal unit overlay management routines, which allow units to be swapped between main memory and disk storage during program execution
`Printer`	Provides easy access to a printer connected to your computer system by declaring a `Text` file `LST` and associating it with the DOS device `LPT1`
`System`	Contains run-time support routines for all standard identifiers and is used automatically by any program or unit, without requiring a reference in a `USES` statement

Functions

Some of the predefined functions of Turbo Pascal appear in Table B.3. The functions following the dotted line are not part of standard Pascal.

Table B.3
Table of Functions

Function	Returns
`Abs(num)`	Integer or real absolute value of its integer or real argument.
`Arctan(num)`	Angle whose tangent is *num*. The result is expressed in radians.
`Chr(num)`	Character with ordinal number corresponding to the integer *num*.
`Cos(num)`	Cosine of real angle *num*, expressed in radians.
`EOF(fil)`	Boolean value indicating end of file status of file variable `fil`.
`EOLN(fil)`	Boolean value indicating end of line status of `Text` file `fil`.
`Exp(num)`	Value of *e* (2.71828) raised to the power indicated by its real argument.
`Ln(num)`	Logarithm base *e* of its real argument.
`Odd(num)`	`True` if its integer argument is an odd number; `False` if not.
`Ord(ordinal)`	Ordinal number corresponding to its ordinal type argument.
`Pred(ordinal)`	Predecessor of its ordinal type argument.
`Round(num)`	Closest integer to its real argument.
`Sin(num)`	Sine of real angle *num*, expressed in radians.
`Sqr(num)`	Square of its integer or real argument.
`Sqrt(num)`	Real number representing the positive square root of its integer or real argument.
`Succ(ordinal)`	Successor of its ordinal type argument.
`Trunc(num)`	Integer part of its real argument.

Table B.3
Table of Functions
(*Cont.*)

Function	Returns
`Concat(st1, st2,...,stN)`	String formed by concatenating its argument strings in the order in which they appear.
`Copy(st, pos, num)`	Substring of `st` starting at position `pos` and consisting of `num` characters.
`FileSize(fil)`	Number of components contained in its file argument.
`Frac(num)`	Fractional part of its real argument `num`.
`High(arg)`	If argument is an open array, returns the largest subscript value relative to 0 as the smallest subscript. If argument is an ordinal type, returns the largest value for that type.
`Int(num)`	Real number representing the whole number part of its real argument.
`IOResult`	Number of input/output error (returns 0 if no input/output error has occurred since previous call).
`Length(st)`	Number of characters in its string argument `st`.
`Low(arg)`	Returns the smallest value for its ordinal type argument.
`New(ptype, constructor)`	A pointer to object storage is allocated on the heap.
`Pi`	Approximation to `Pi` (3.1415926536).
`Pos(substr, st)`	Starting position in `st` of first occurrence of the string contained in `substr` (returns 0 if `substr` does not appear in `st`).
`Random or Random(int)`	Real random number between 0.0 and 1.0 if no argument given; if integer argument is given, returns random integer greater than or equal to 0 and less than `int`. The procedure `Randomize` should be called prior to the first reference to `Random`.
`UpCase(ch)`	Uppercase equivalent of `Char` argument `ch`, if one exists.

Procedures

Some of the predefined procedures of Turbo Pascal appear in Table B.4. The procedures following the dotted line are not part of standard Pascal.

Table B.4
Table of Procedures

Procedure	Effect
`Dispose (p)`	Returns dynamic storage pointed to by pointer variable `p` to heap.
`New (p)`	Creates new dynamic variable and sets pointer variable `p` to point to its memory location.
`Read (f, variables)`	Reads data from file `f` to satisfy the list variables. If `f` is not a `Text` file, only one component can

Table B.4
Table of Procedures
(Cont.)

Procedure	Effect
	be read at a time. If f is not specified, data are read from file Input (the keyboard).
ReadLn (f, variables)	Reads data from Text file f to satisfy the list of variables; skips any characters at the end of the last line read.
Reset (f)	Opens file f for input and sets the file-position pointer to the beginning.
Rewrite (f)	Prepares file f for output and sets the file-position pointer to the beginning. Prior file contents are lost.
Write (f, outputs)	Writes data in the order specified by outputs to file f. If f is not a Text file, only one component may be written at a time. If f is not specified, data are written to Output (the screen).
WriteLn (f, outputs)	Writes data in order specified by outputs to Text file f; writes end-of-line marker after the data.
Assign (f, st)	Assigns name of external file contained in the string expression st to file variable f.
Close (f)	Closes file f.
Delete (st, pos, num)	Removes substring of string st starting at position pos and consisting of num characters.
Dispose (p, destructor);	If called with a destructor as second argument, Dispose can be used to return object storage to heap.
Erase (f)	Erases external file associated with file variable f from disk.
Exit	Halts execution of current block and returns control to the calling block.
Halt	Stops program execution and returns control to the operating system.
Insert (obj, targ, pos)	Inserts string obj into string targ starting at position pos in targ.
New (p, constructor)	If called with a constructor as second argument, New can be used to allocate and initialize heap storage for an object.
Randomize	Initializes the built-in random-number generator with a random value derived from the system clock.
Seek (f, recnum)	Moves file-position pointer for file f to component number indicated by LongInt argument recnum.
Str (numval, st)	Converts numeric value of numval to string stored in st. Form of st is specified by format part of numval.
Val (st, num, code)	If successful, converts string st to an integer or real value as determined by the type of num and code is set to 0. If not successful, code will be set to the position of first offending character in st.

Compiler Directives

A Turbo Pascal compiler directive consists of an opening curly brace ({) followed by a dollar sign, followed by the option name (one or more letters), followed by the option value (+ or –) or parameters affecting the compilation of the program or unit, and is terminated by a closing curly brace (}). Spaces are not allowed before a dollar sign or between the option name and option value. At least one space must separate the option name from a parameter. Examples of several compiler directives appear below.

```
{$B-}
{$R+}
{$B-, $R+, $D-}     {3 compiler directives}
```

A plus sign as the value of a compiler option enables it (makes it active), and a minus sign value disables the option (makes it passive). The compiler directive

```
{$I INCLUDE.PAS}
```

includes the source code from file `INCLUDE.PAS` during compilation.

Table 8.5 Compiler Option Directives

Directive	Default	Effect
Align Data	{$A+}	Align variables on word boundaries.
Boolean Evaluation	{$B-}	Use short-circuit evaluation of Boolean expressions.
Debug Information	{$D+}	Generate debug information during compilation. Usually used with {$L+}.
Emulation	{$E+}	Links floating-point run-time library, which emulates the 80x87 numeric coprocessor.
Force Far Call	{$F-}	Allow Turbo Pascal to choose near or far call model for function and procedure calls, based on program context.
Generate 286 Instructions	{$G-}	Do not use any special 80286 processor instructions during code generation.
Input/Output Checking	{$I+}	Enables automatic generation of code to check the result of an input/output procedure call.
Local Symbol Information	{$L+}	Generate local symbol information during compilation. Must be used with {$D+}.
Numeric Processing	{$N-}	Perform all real-type calculations by calling the TP run-time library routines and not the actual 80x87 routines.
Overlay Code Generation	{$O-}	Disables overlay code generation.
Open Parameters	{$P-}	Disallows use of open string or array as declarations for function or procedure formal parameters.

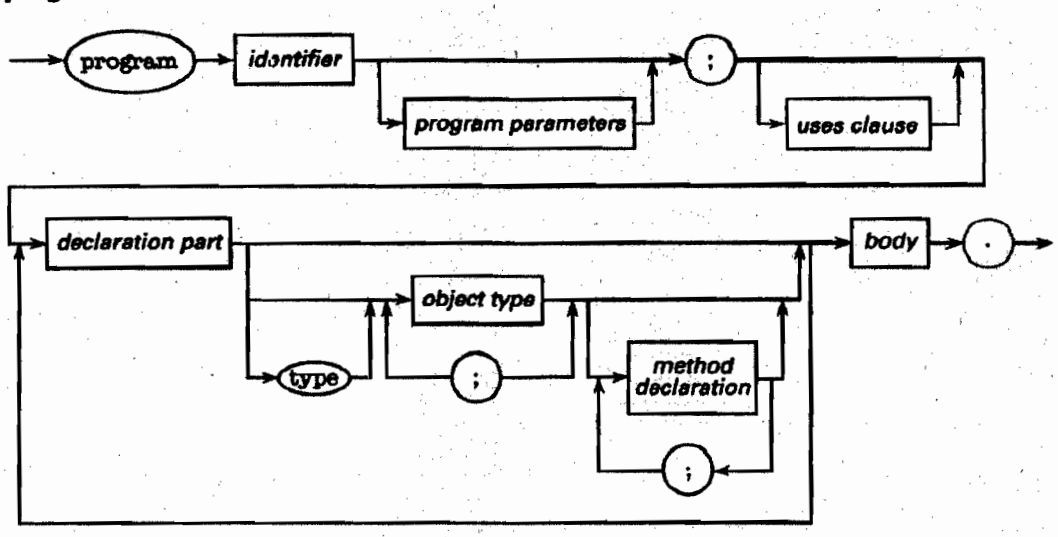
Table B.5
Compiler Option Directives
(Cont.)

Directive	Default	Effect
Overflow Checking	{$Q-}	Disables generation of code to check for integer overflow during arithmetic operations. {$Q+} is often used with {$R+}.
Range Checking	{$R-}	Disables generation of code to check for range-checking and object initialization violations.
Stack-Overflow Checking	{$S+}	Enables generation of code at beginning of each procedure or function, which checks for enough stack space to meet local data needs of subprogram.
Typed @ Operator	{$T-}	Disables pointer type checking for @ result value.
Var-String Checking	{$V+}	Enables strict type checking for string variables passed to `var` parameters.
Extended Syntax	{$X+}	Enables support for special Turbo Pascal function and string capabilities.
Symbol Reference Information	{$Y+}	Enables generation of symbol reference information.

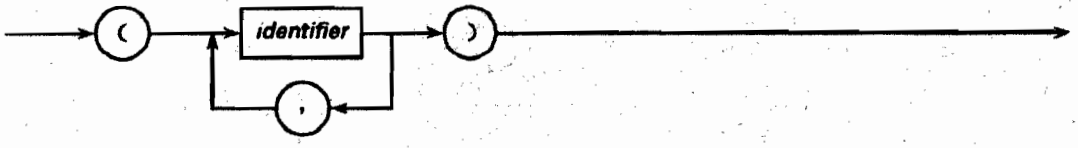
ภาคผนวก B

Turbo Pascal Syntax Diagrams

program

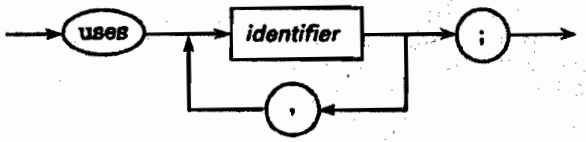


program parameters

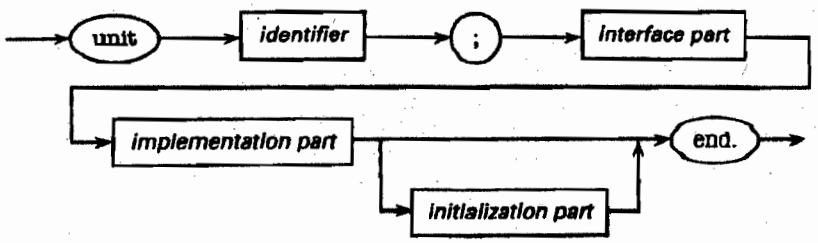


body

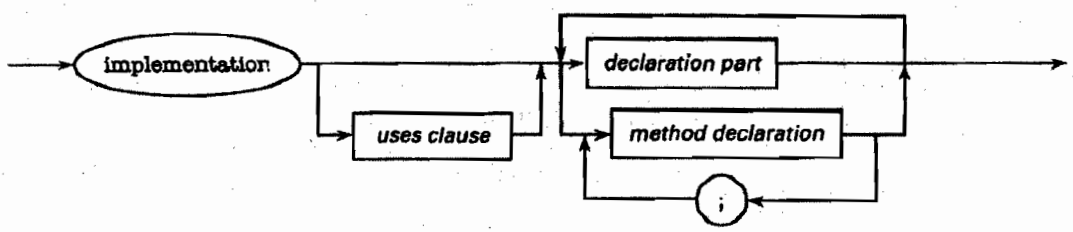
uses clause



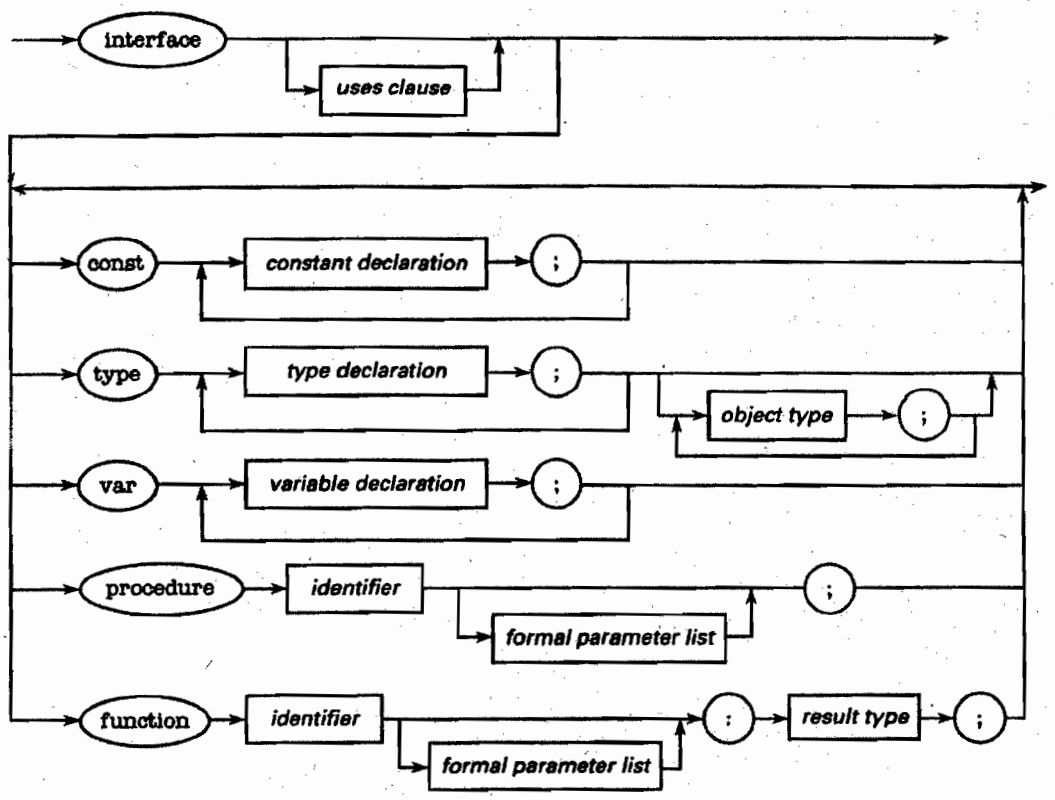
unit



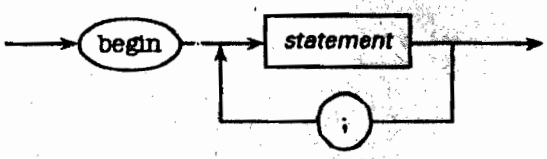
implementation part



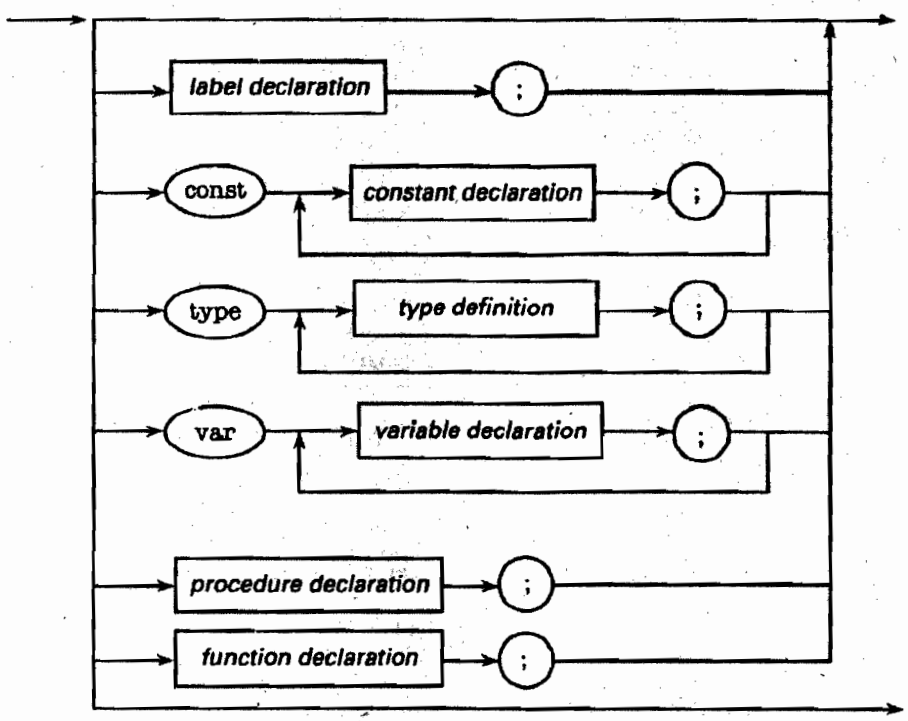
interface part



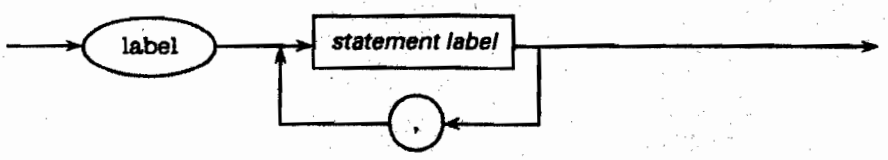
initialization part



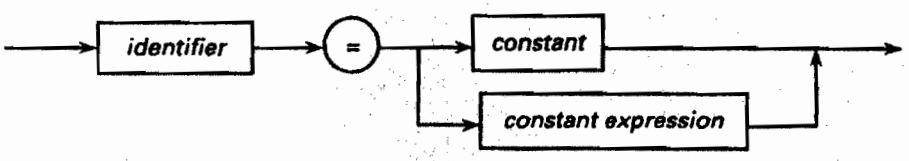
declaration part



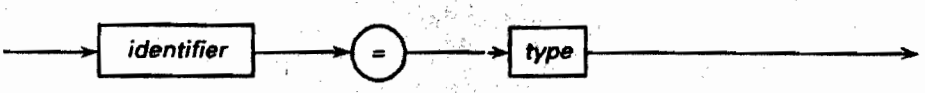
label declaration



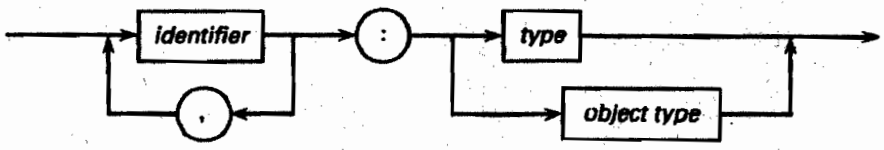
constant definition



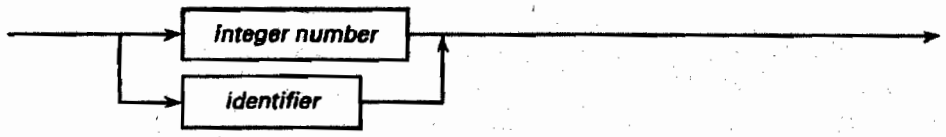
type definition



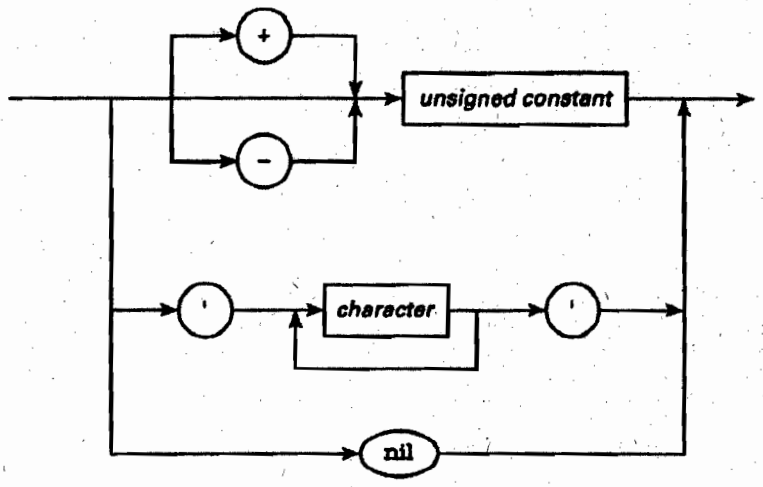
variable declaration



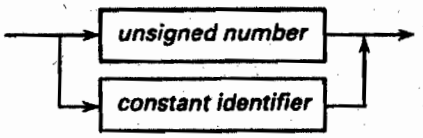
statement label



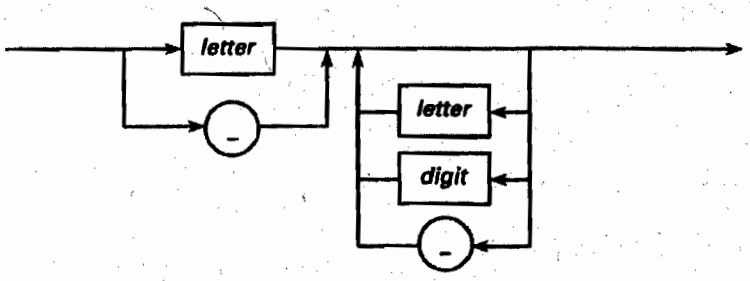
constant



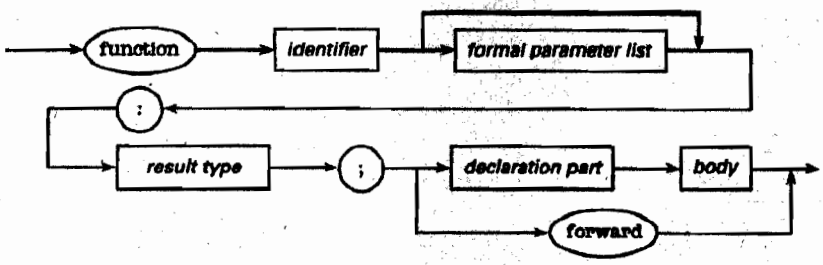
unsigned constant



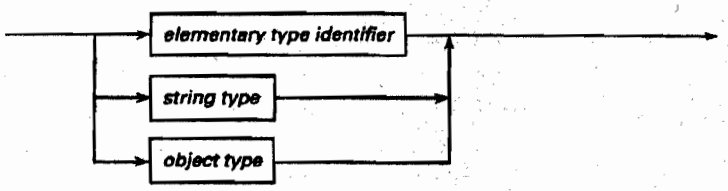
identifier



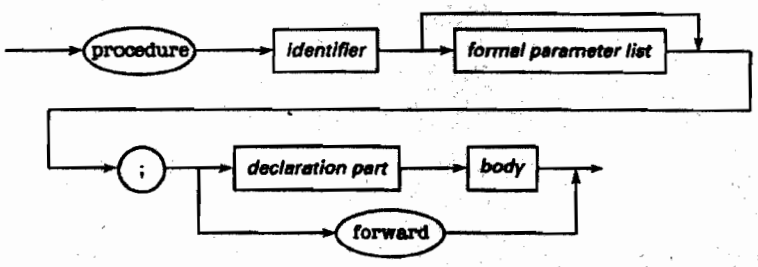
function declaration



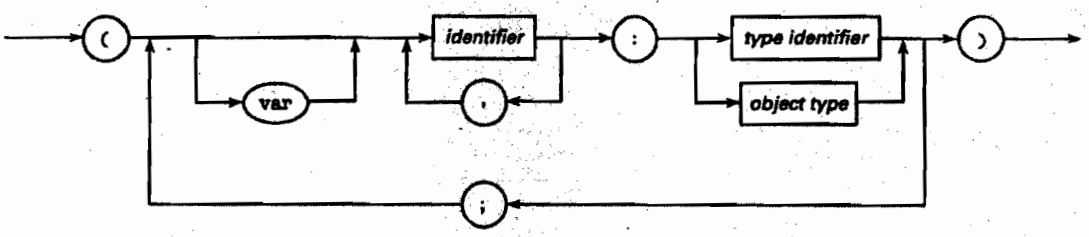
result type



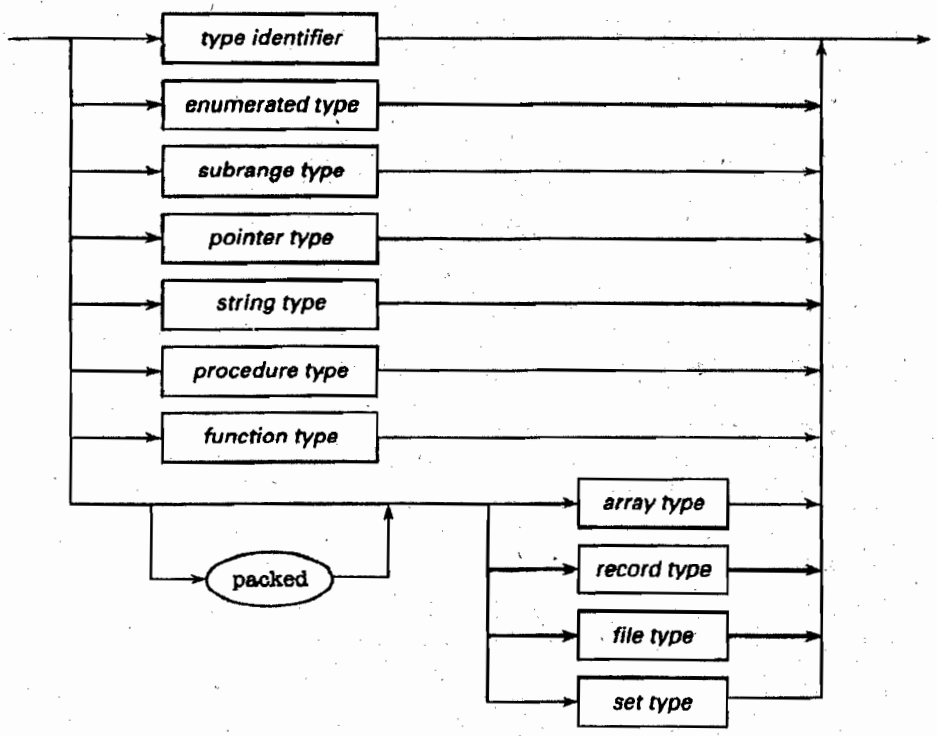
procedure declaration



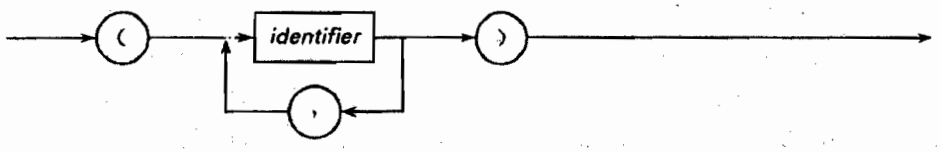
formal parameter list



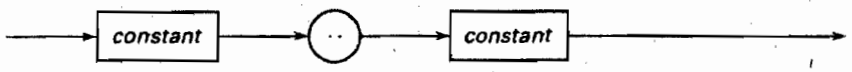
type



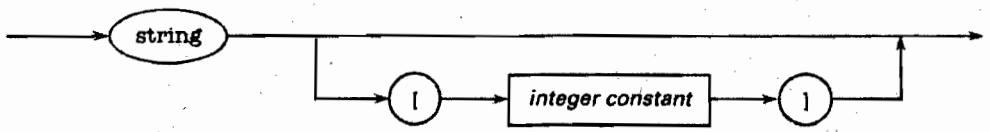
enumerated type



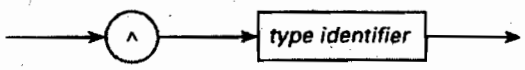
subrange type



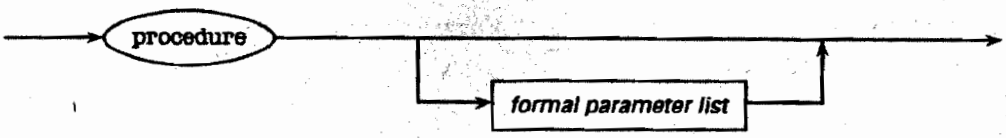
string type



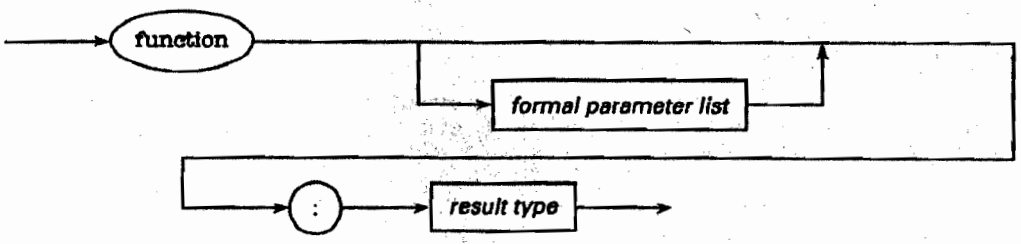
pointer type



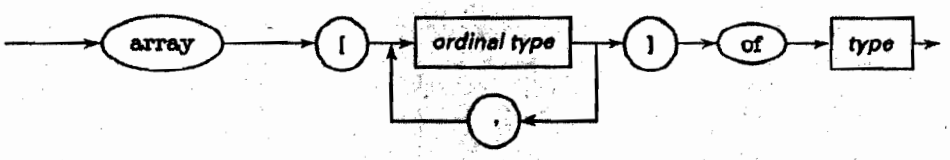
procedure type



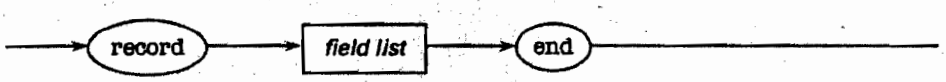
function type



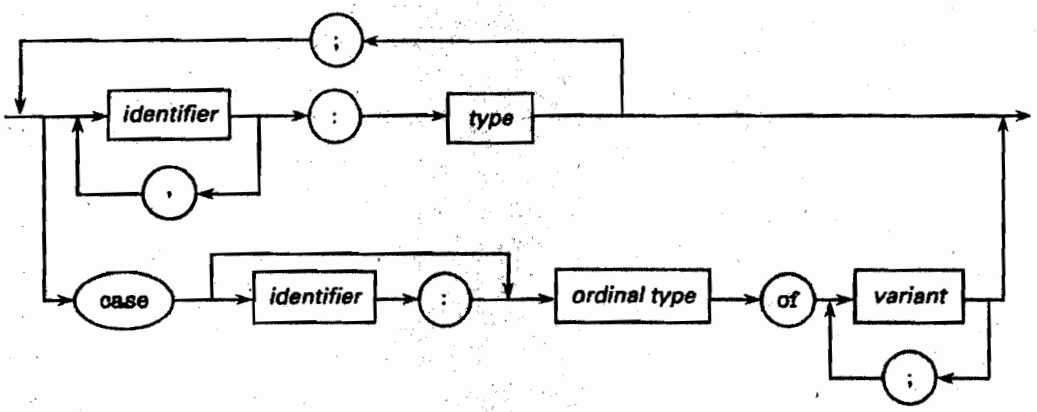
array type



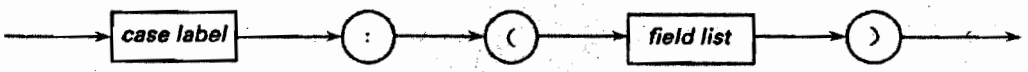
record type



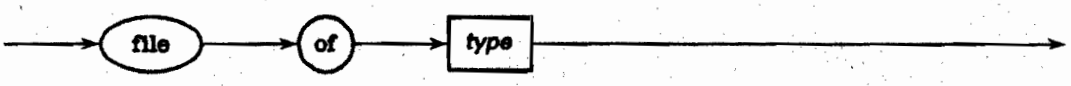
field list



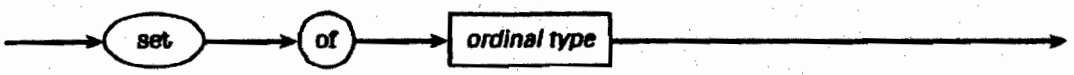
variant



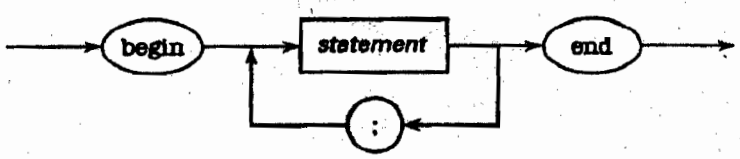
file type



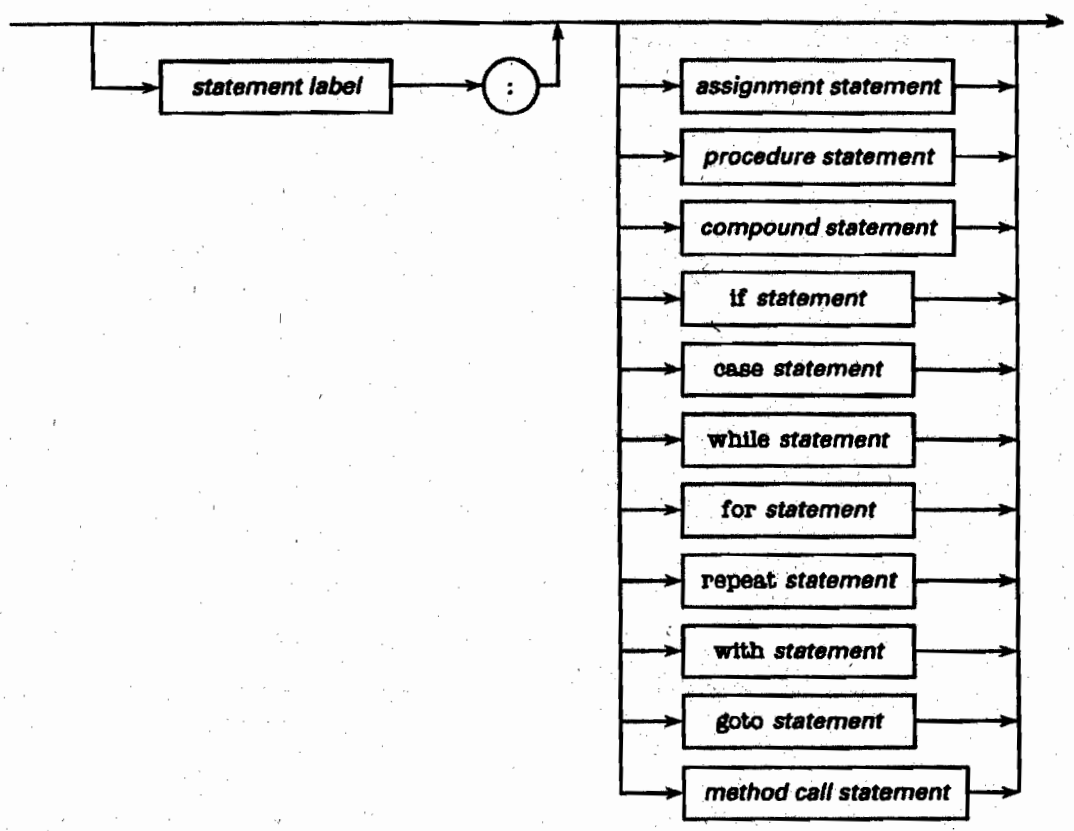
set type



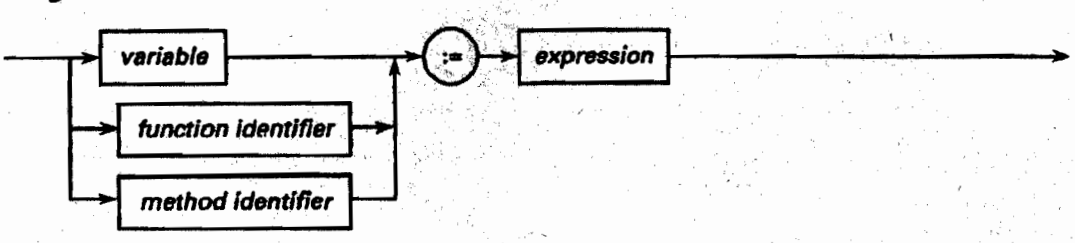
compound statement



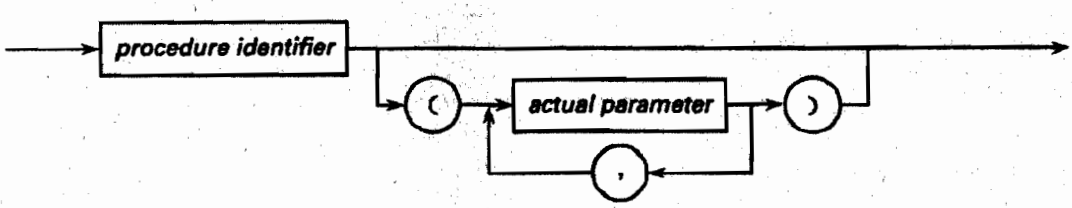
statement



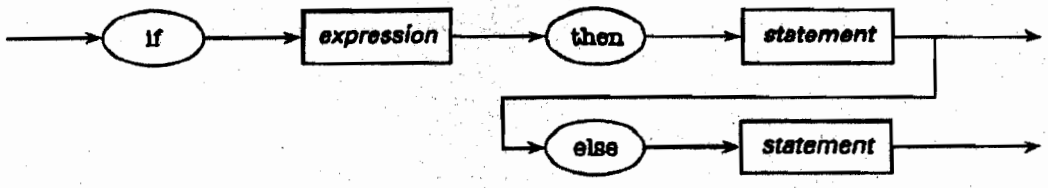
assignment statement



procedure call statement

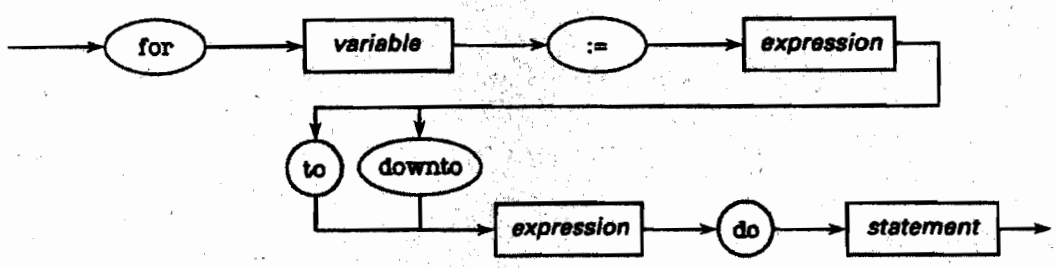


if statement

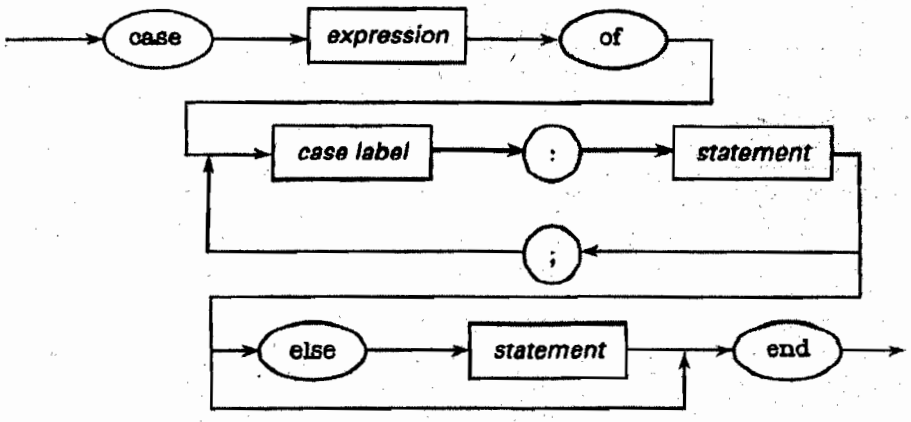


while statement

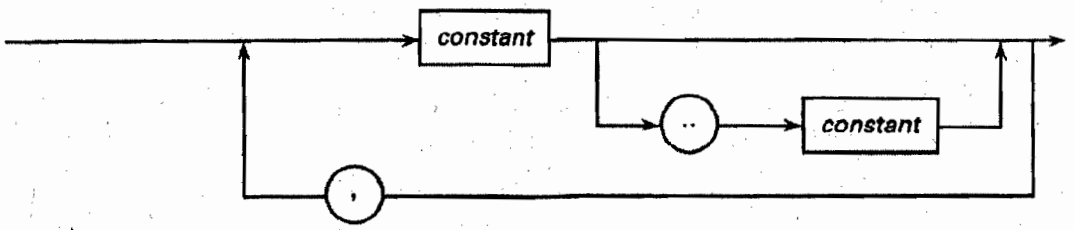
for statement



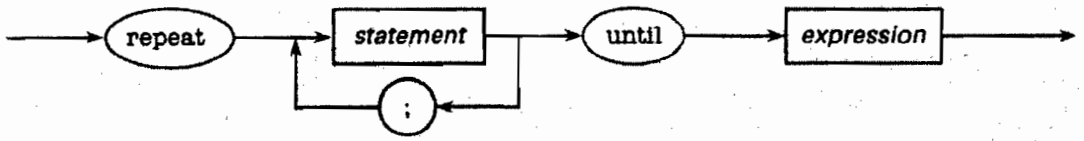
case statement



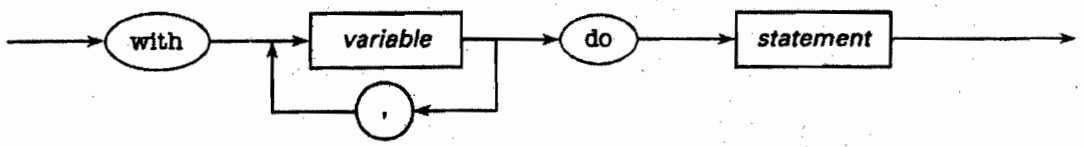
case label



repeat statement

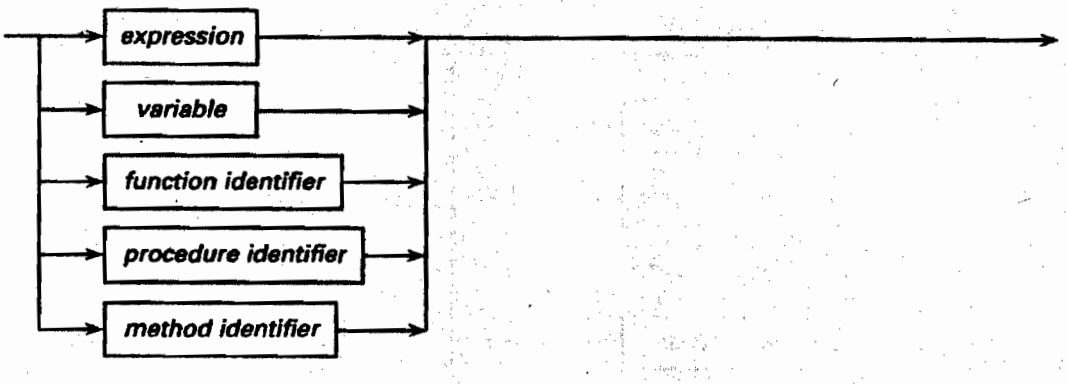


with statement

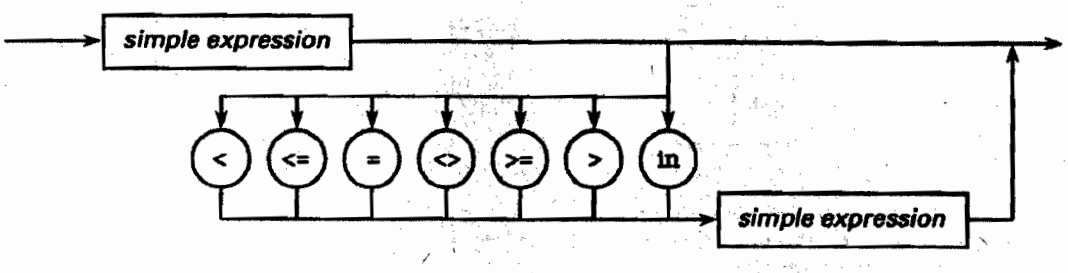


goto statement

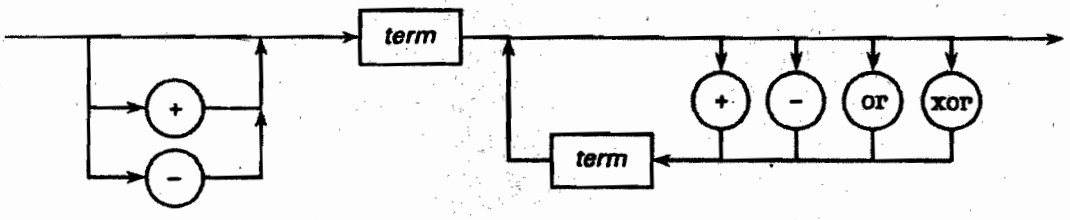
actual parameter



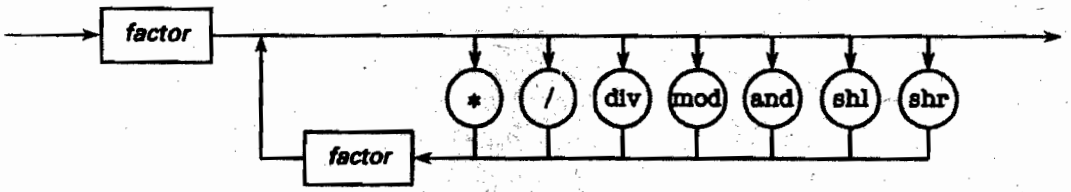
expression



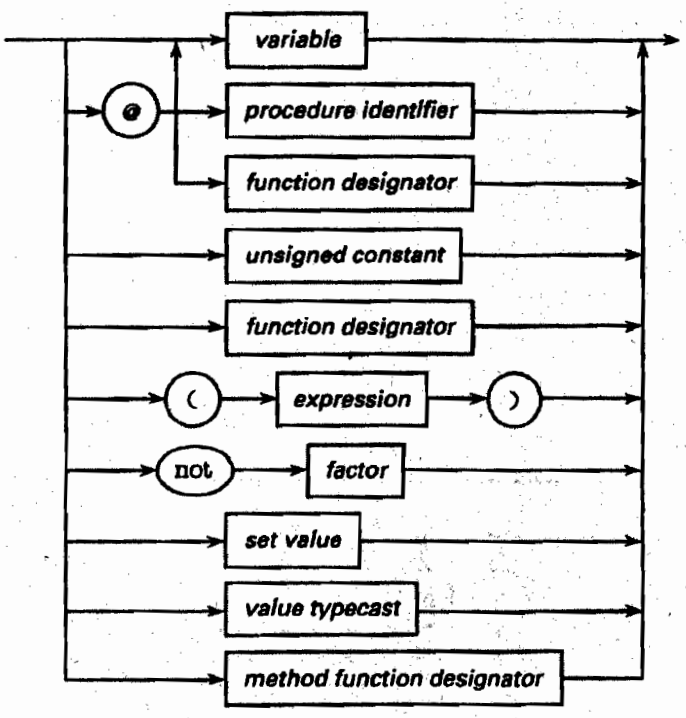
simple expression



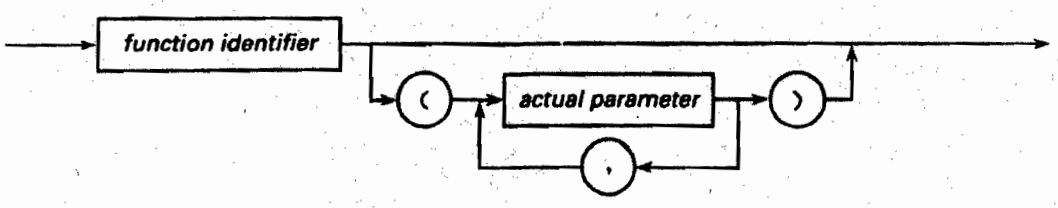
term



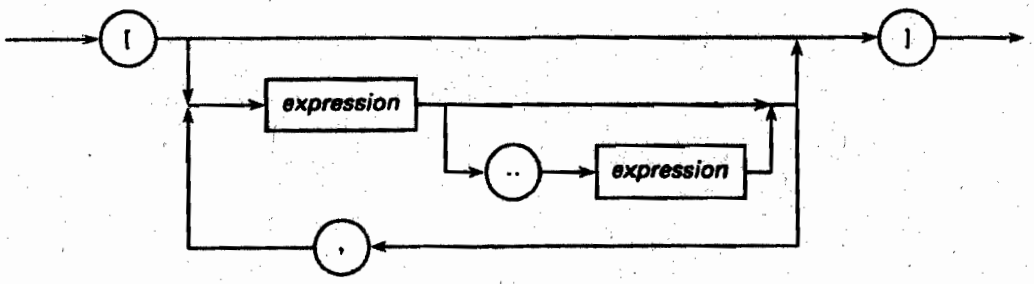
factor



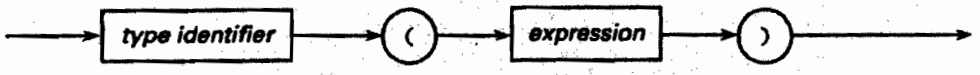
function designator



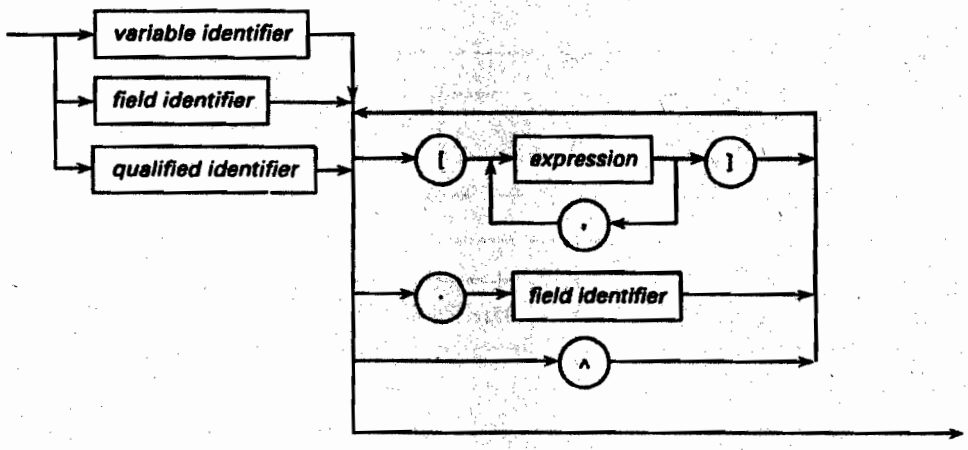
set value



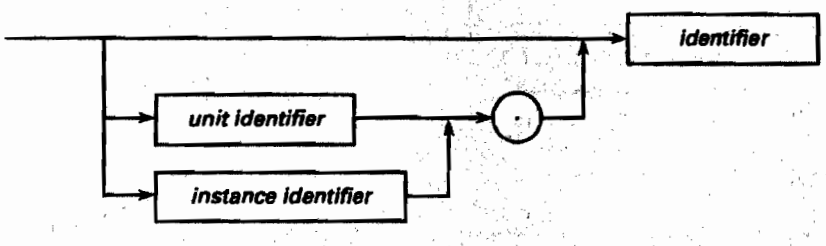
value typecast



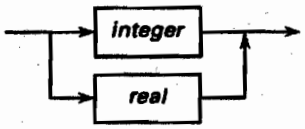
variable



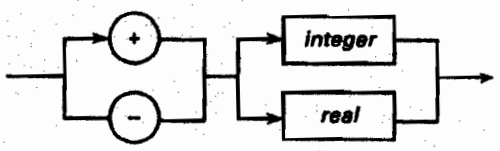
qualified identifier



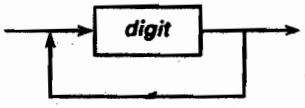
unsigned number



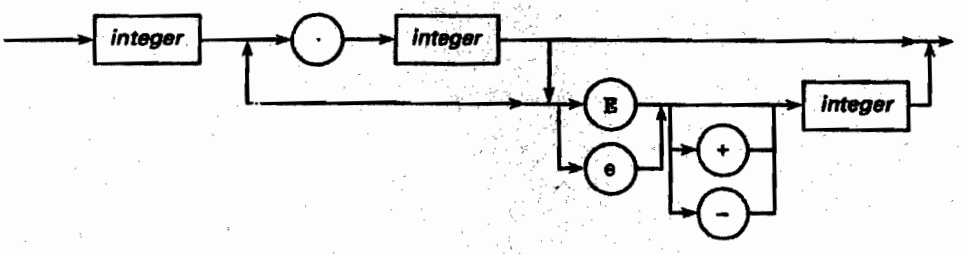
signed number



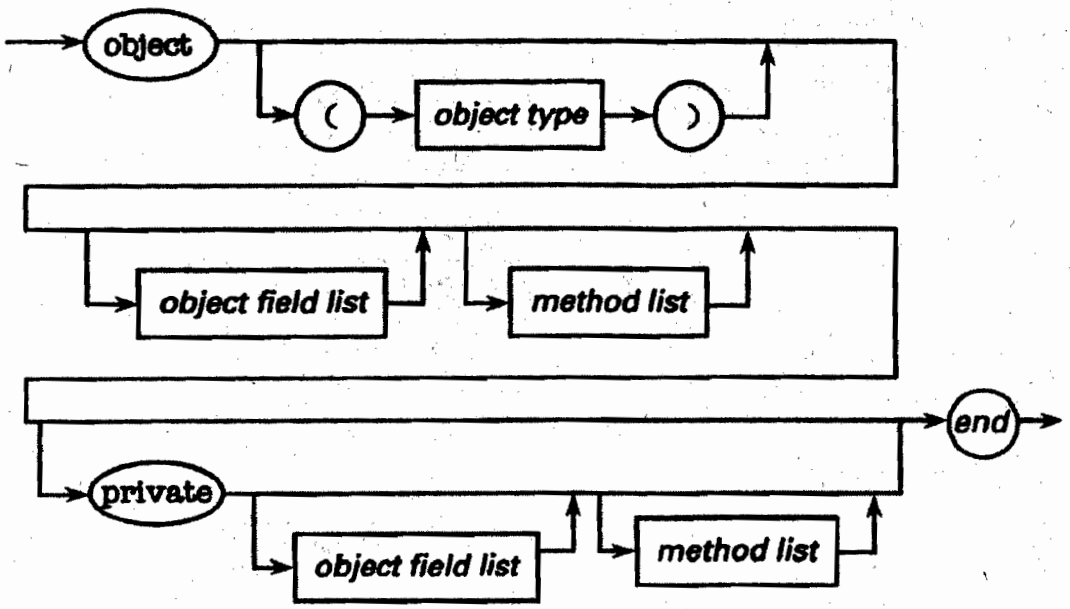
integer



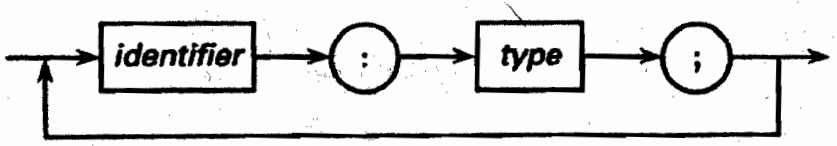
real



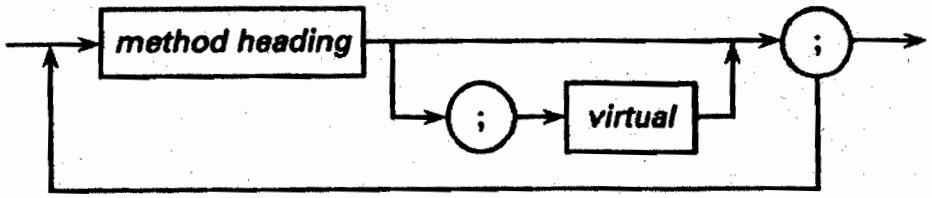
object type



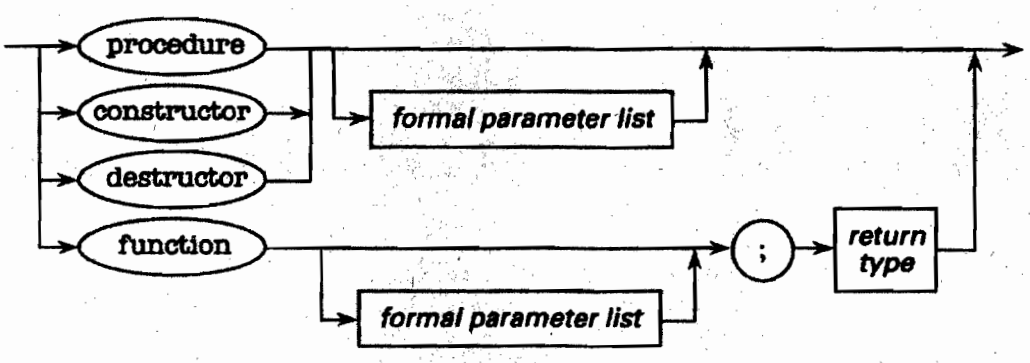
object field list



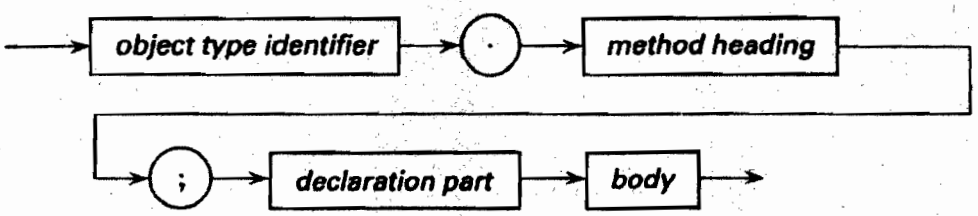
method list



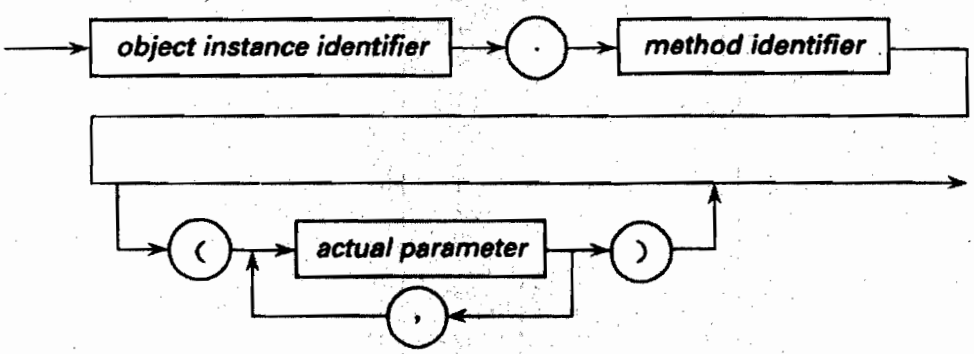
method heading



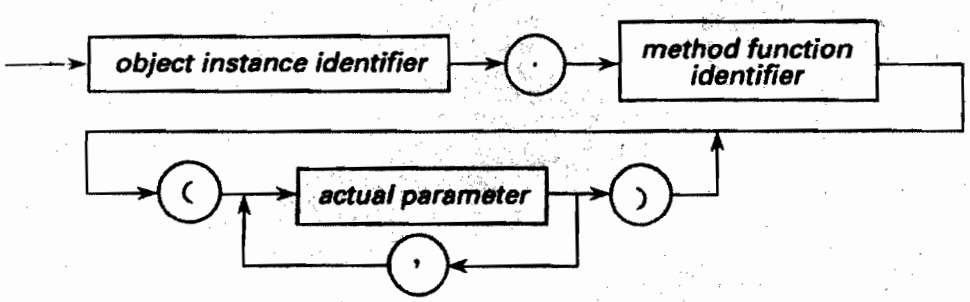
method declaration



method call statement



method function designator



ASCII Character Set

Table D.1 contains the character codes from 0 to 127 for Turbo Pascal. On most computer systems, only the codes from 32 (blank or space) to 126 (symbol ~) have printable characters. The other codes represent the nonprintable control characters. The IBM PC implementation of Turbo Pascal provides printable symbols for the nonprintable codes as well.

Table D.1
Table of ASCII Characters

Code	Char	Code	Char	Code	Char	Code	Char
0	^@ NUL	32	□	64	@	96	`
1	^A SOH	33	!	65	A	97	a
2	^B STX	34	"	66	B	98	b
3	^C ETX	35	#	67	C	99	c
4	^D EOT	36	$	68	D	100	d
5	^E ENQ	37	%	69	E	101	e
6	^F ACK	38	&	70	F	102	f
7	^G BEL	39	'	71	G	103	g
8	^H BS	40	(	72	H	104	h
9	^I HT	41	)	73	I	105	i
10	^J LF	42	*	74	J	106	j
11	^K VT	43	+	75	K	107	k
12	^L FF	44	,	76	L	108	l
13	^M CR	45	-	77	M	109	m
14	^N SO	46	.	78	N	110	n
15	^O SI	47	/	79	O	111	o
16	^P DLE	48	0	80	P	112	p
17	^Q DC1	49	1	81	Q	113	q
18	^R DC2	50	2	82	R	114	r
19	^S DC3	51	3	83	S	115	s
20	^T DC4	52	4	84	T	116	t
21	^U NAK	53	5	85	U	117	u
22	^V SYN	54	6	86	V	118	v
23	^W ETB	55	7	87	W	119	w
24	^X CAN	56	8	88	X	120	x
25	^Y EM	57	9	89	Y	121	y
26	^Z SUB	58	:	90	Z	122	z
27	^[ESC	59	;	91	[	123	{
28	^\ FS	60	<	92	\	124	|
29	^] GS	61	=	93	]	125	}
30	^^ RS	62	>	94	^	126	~
31	^_ US	63	?	95	_	127	DEL