12 ZÁVĚR

V této poslední kapitole se budeme zabývat příkazem, jehož užívání v programech není nutné a může vést k nepřehledným programům. Proto často bývá začátečníkům zamlčován. Rozumným užíváním však může tento příkaz program zefektivnit bez ztráty přehlednosti.

12.1 Příkaz skoku

V kap. 4 jsme hovořili o řídicích strukturách. V Pascalu jsou všechny řídicí struktury navrženy tak, že mají tvar jednoho příkazu. Program je zapsán přehledně, ale může se stát, že není dostatečně efektivní. V některých případech si totiž může užívání popsaných struktur vynutit zavedení pomocných logických proměnných, popř. vícenásobných testů téže podmínky.

Kromě uvedených řídicích struktur existuje v Pascalu ještě jedna možnost řízení běhu programu. Je nazvána *skok*.

Příkazy programu zapsané za sebou v textu programu byly dosud v našich programech prováděny v pořadí, v němž byly v textu napsány. Skok umožňuje toto pořadí prováděných příkazů změnit.

V každém bloku je možné libovolný příkaz označit celým číslem nazvaným *návěští*:

návěští : *příkaz*

Lze tedy napsat

```
12 : I := I + 1;
35 : while A < 1 do A := A - 2.3;
```

Na takto označený příkaz lze přesunout řízení (pokračování výpočtu) při běhu programu *příkazem skoku* (go to = jdi na).

goto *návěští*

např.

goto 12

Příkaz skoku může program urychlit. V mnoha případech ovšem také může program znepřehlednit, což je daleko větší nedostatek (zejména pro začá-

tečníky). Proto se užívání příkazu skoku, pokud je to možné, vyhýbáme a klademe na ně jistá omezení.

Všechna návěští deklarovaná u příkazů v daném bloku musí být definována v části definice návěští v témže bloku a každé definované návěští musí být použito:

label
návěští$_1$, ... , *návěští*$_n$;

Část definice návěští se nachází na začátku bloku před všemi deklaracemi.

label
návěští$_1$, ... , *návěští*$_n$;
const
:
type
:
var

Dále příkaz skoku nesmí směřovat zvnějšku do řídicí struktury. Není povoleno např.

```
while A < 0 do
  begin
    :
    28 : A := A – 2.3;
    :
  end;
:
goto 28;
```

Naopak je povolen skok z řídicí struktury, a dokonce i skok z procedury nebo funkce do bloku nadřazené funkce nebo procedury, např.

```
procedure A;
label 1;
  procedure B;
  begin { tělo procedury B }
    :
    goto 1;
    :
  end; { B }
begin
  :
  1: I := I+1;
  :
end; { A }
```

V některých algoritmech je využití příkazu skoku opodstatněné, i když není nutné. Jsou to zejména skoky ukončující okamžitě provedení řídicí struktury. Například skok na konec těla bloku

```
    begin { tělo A }
      :
      goto 999;
      :
999:  end; { A }
```

popř. na konec některého z nadřazených bloků (nebo i na konec programu). Časté je také užití příkazu skoku pro předčasné ukončení cyklu.

Chceme-li např. prohledat pole celočíselných složek

```
var
  A : array [1..100] of integer;
```

nalézt v něm první nenulovou hodnotu a její index uložit do proměnné PRVNI, je možné napsat

```
label 100;
var
  I       : integer;
  A       : array [1..100] of integer;
  PRVNI   : integer;
begin
  :
  PRVNI := 0; { hodnota 0 znamená nenalezení }
  for I := 1 to 100 do
    if A[I] <> 0 then
      begin { nenulová složka nalezena }
        PRVNI := I;
        goto 100;
      end; { if }
100: { pokračování programu }
  :
end;
```

12.2 Cvičení

1. Program 11.2 (editor) modifikujte tak, že po zadání příkazu E (konec programu) se nenastaví proměnná CYKLUJ na hodnotu *false*, nýbrž se provede skok na konec programu. Podobně lze ukončovat i další konverzační programy.

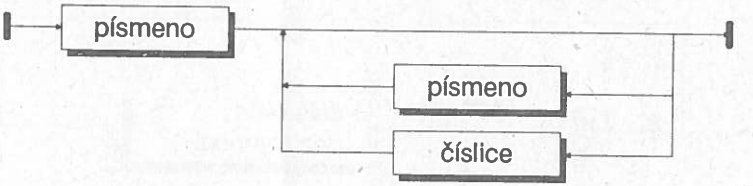
A SYNTAKTICKÁ DEFINICE PASCALU

Všechny zápisy v programu mají přesně definovaný tvar. Říkáme, že je definována jejich skladba, neboli *syntaxe*. Slovní popis syntaxe prvků programu není často dostatečně přesný, proto zavedeme jednoduché grafické prostředky pro vyjádření skladby jednotlivých konstrukcí jazyka. Říkáme jim *syntaktické diagramy*.

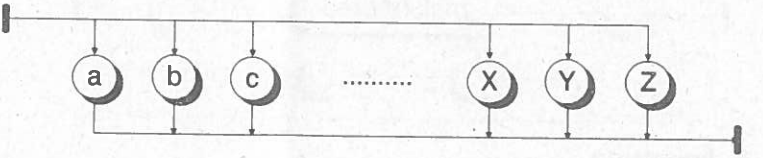
Každý syntaktický diagram popisuje jednu konstrukci Pascalu. Její jméno je vždy uvedeno vlevo nad příslušným diagramem. Syntaktický diagram určuje všechny povolené tvary konstrukce. Každý povolený tvar získáme tak, že projdeme uzly od začátku do konce diagramu. Začátek diagramu je vždy vlevo nahoře.

Například syntaktický diagram definující identifikátor má tvar:

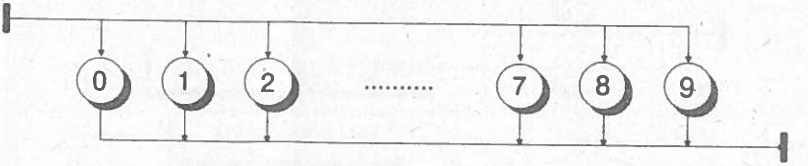
- **identifikátor**



- **písmeno**



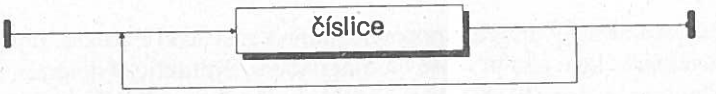
- **číslice**



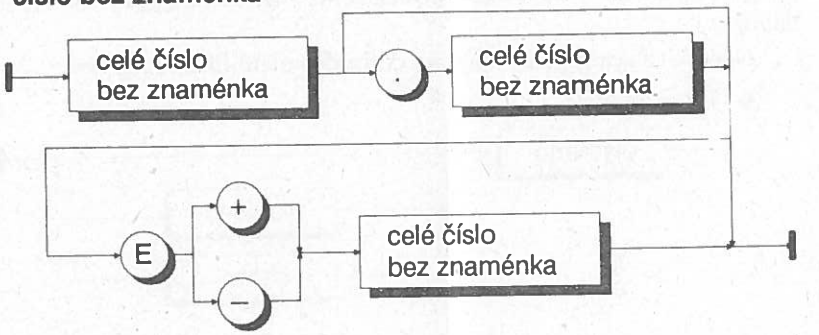
Posloupnost symbolů tvořících správnou konstrukci Pascalu vytváříme zleva doprava :

- Procházíme-li uzlem diagramu, který má tvar kruhu nebo oválu, pak symbol v uzlu přidáme do posloupnosti.
- Procházíme-li uzlem ve tvaru obdélníka (znamená odkaz na jiný vnořený syntaktický diagram), pak k vytvářené posloupnosti symbolů přidáme libovolnou posloupnost symbolů definovanou vnořeným diagramem.

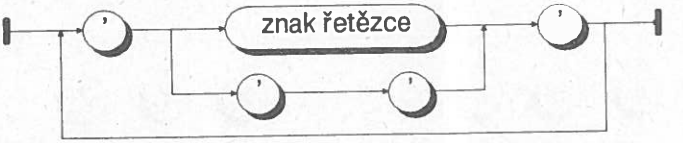
- **celé číslo bez znaménka, návěští**



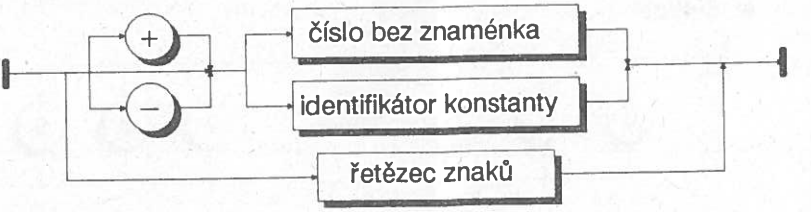
- **číslo bez znaménka**



- **řetězec znaků**

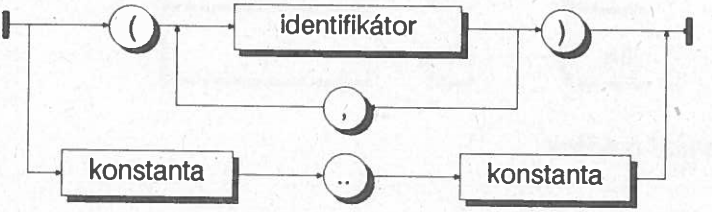


- **konstanta**

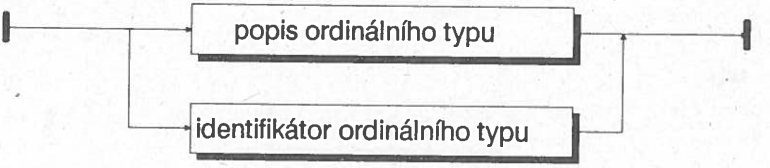


- **konstanta bez znaménka**

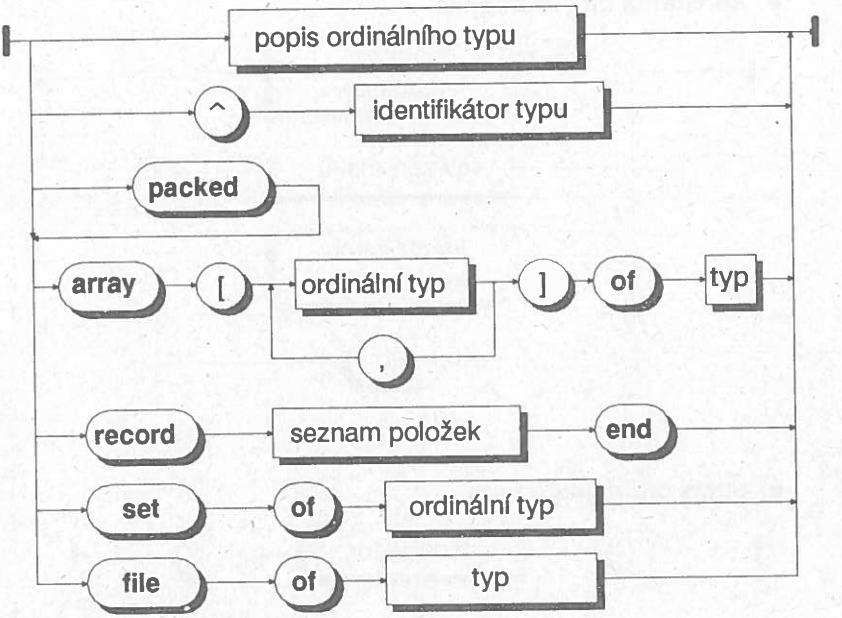
- **popis ordinálního typu**



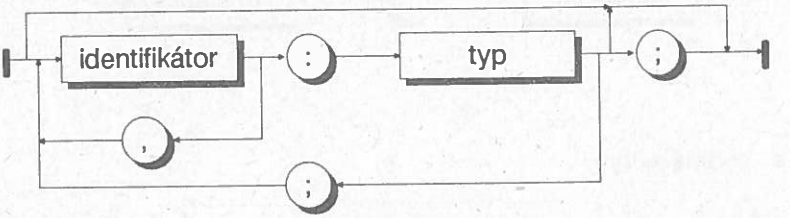
- **ordinální typ**



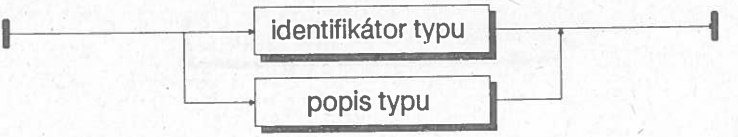
- **popis typu**



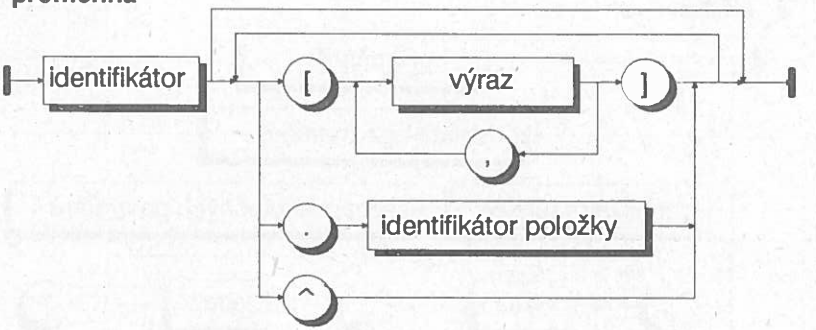
- **seznam položek**



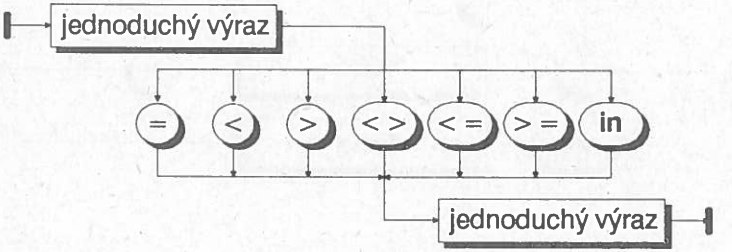
- **typ**



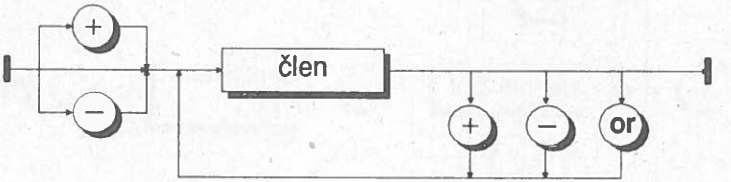
- **proměnná**



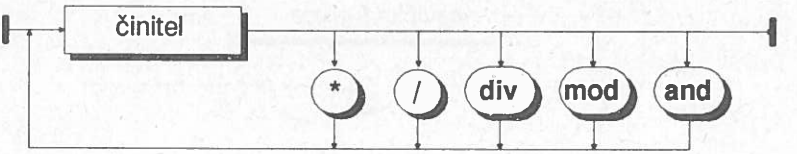
- **výraz**



- **jednoduchý výraz**

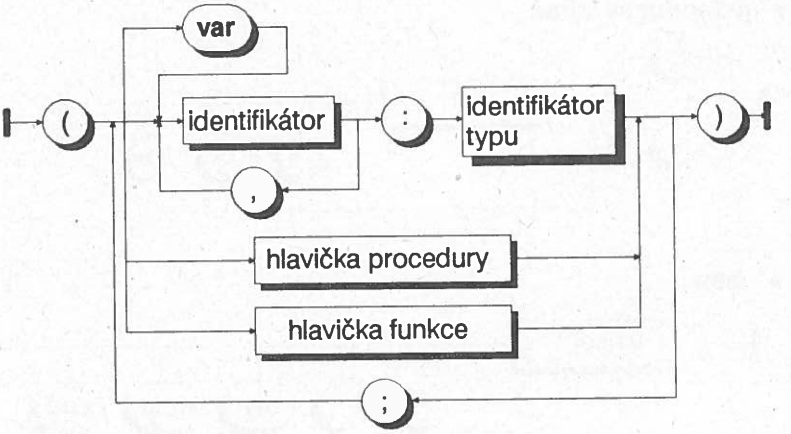


- **člen**

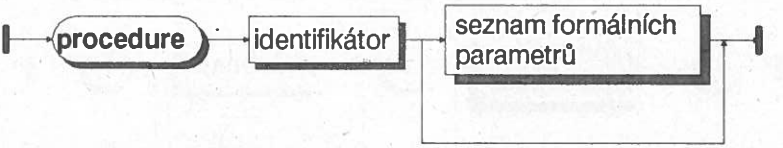


- **činitel**

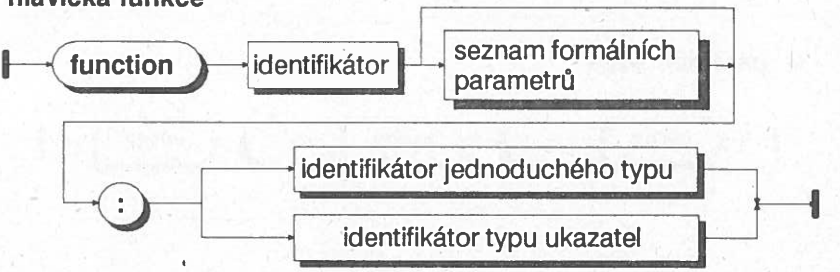
- **seznam formálních parametrů**



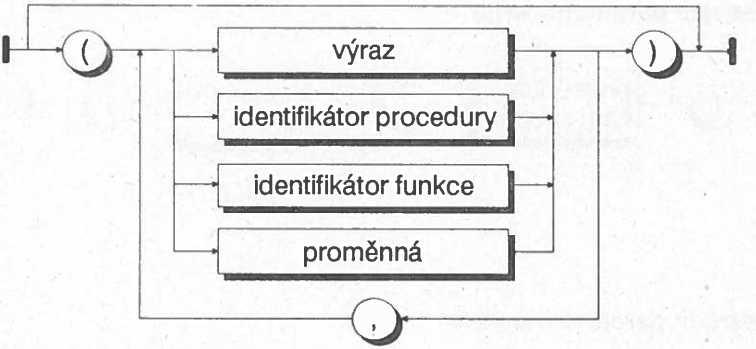
- **hlavička procedury**



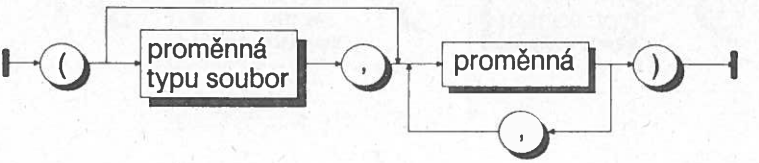
- **hlavička funkce**



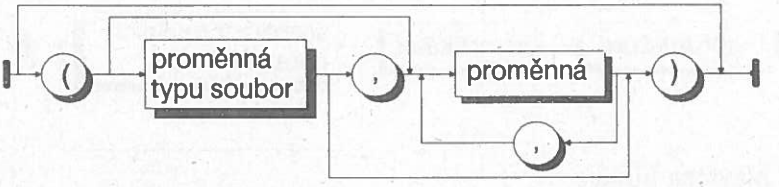
- **seznam skutečných parametrů**



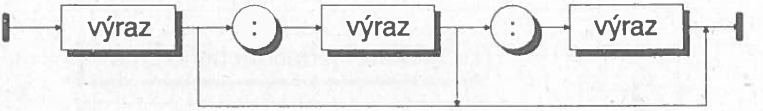
- **seznam parametrů read**



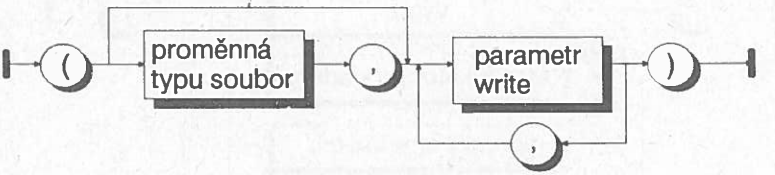
- **seznam parametrů readln**



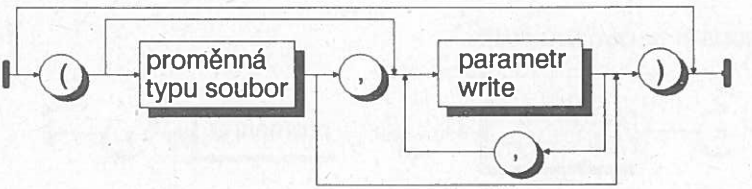
- **parametr write**



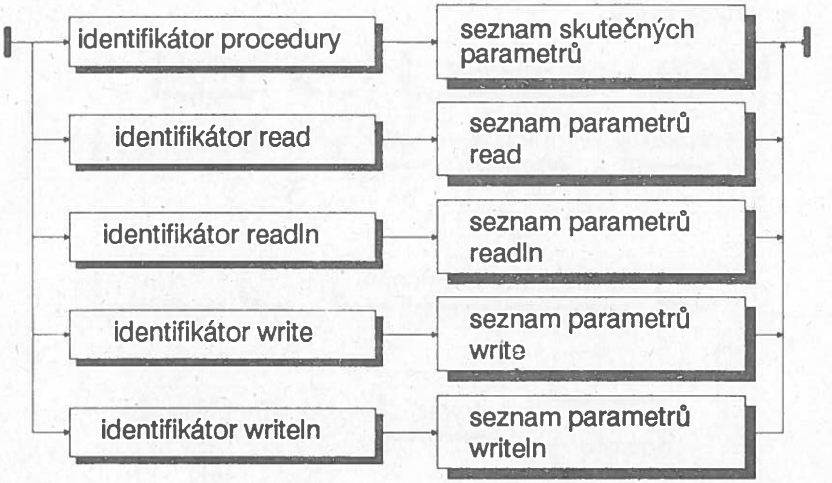
- **seznam parametrů write**



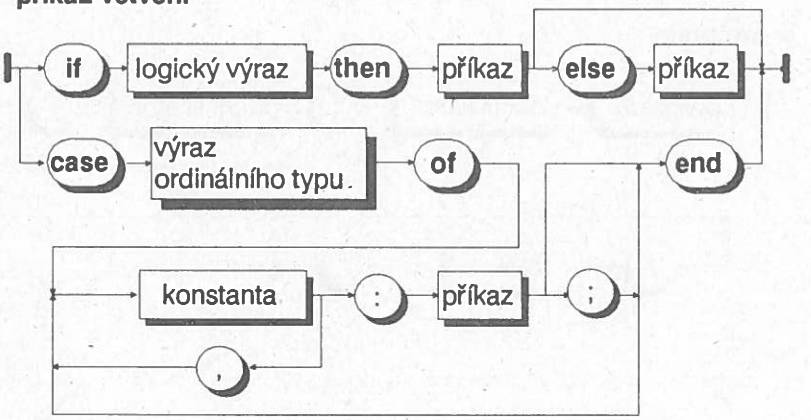
- **seznam parametrů writeln**



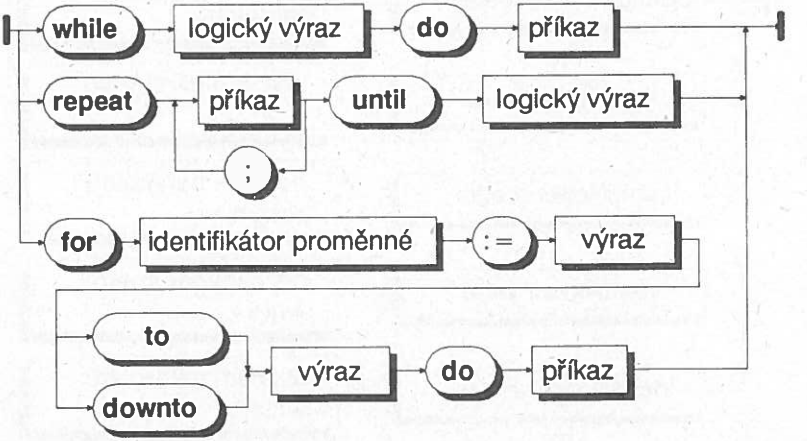
- **příkaz procedury**



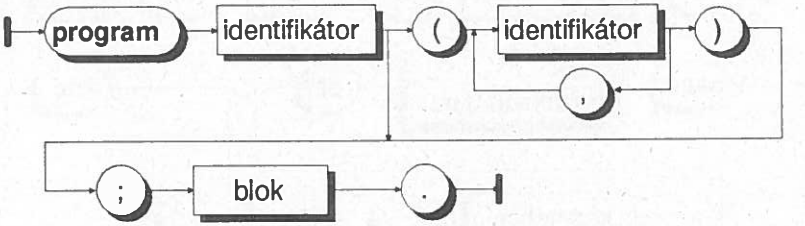
- **příkaz větvení**



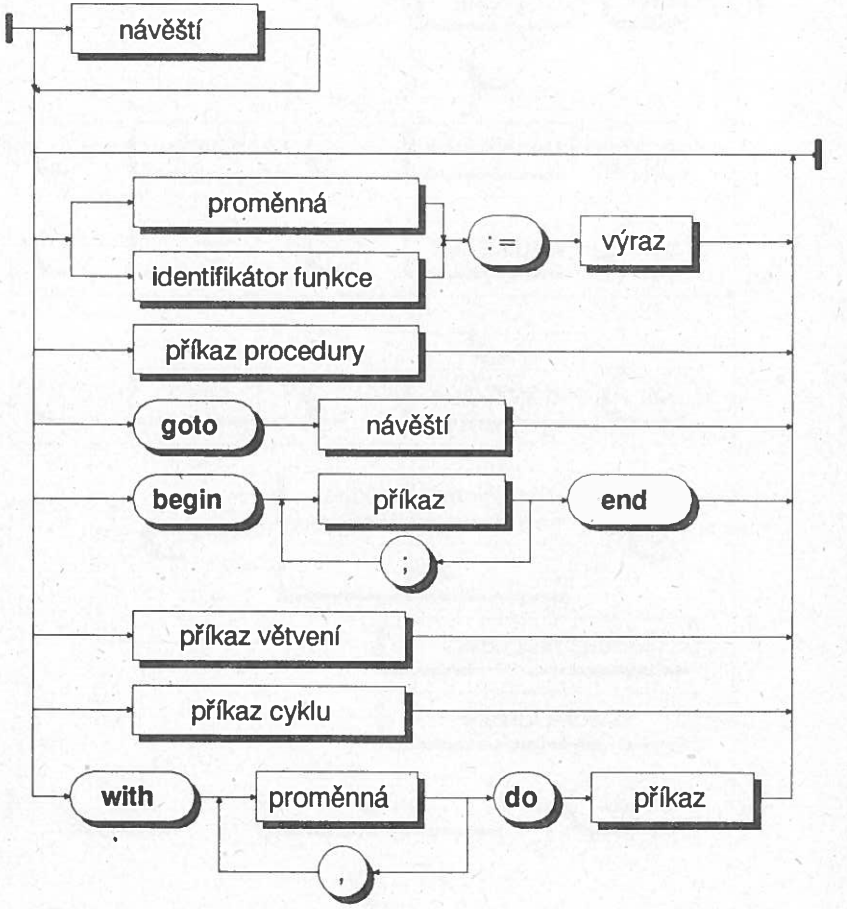
- **příkaz cyklu**



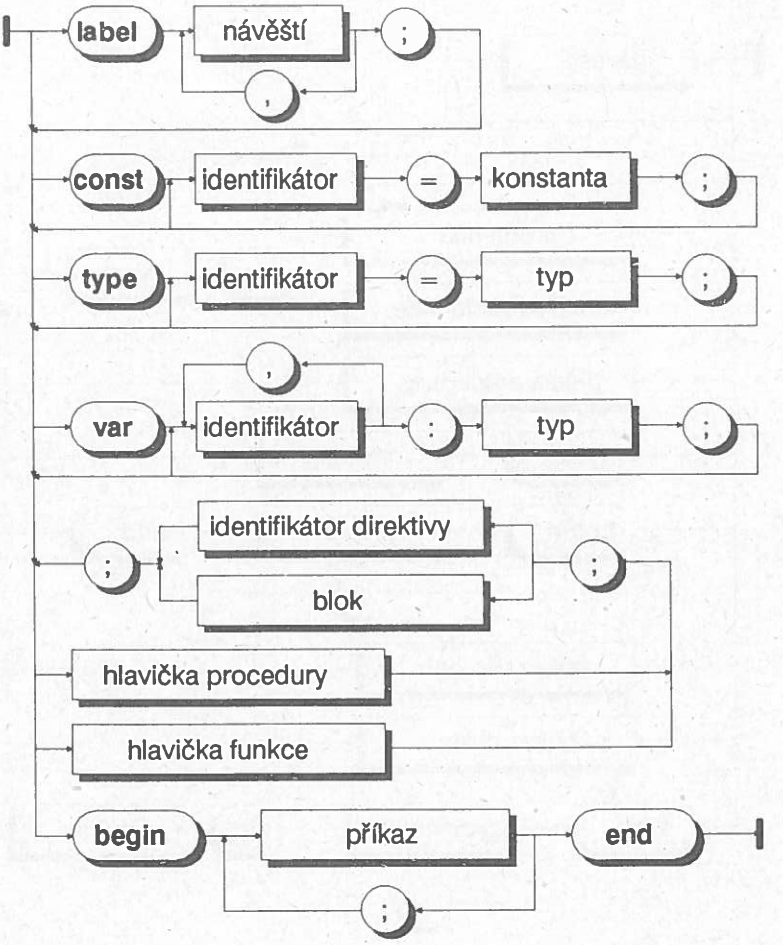
- **program**



- **příkaz**


návěští
proměnná
identifikátor funkce
:=
výraz
příkaz procedury
goto
návěští
begin
příkaz
end
;
příkaz větvení
příkaz cyklu
with
proměnná
do
příkaz
,

- **blok**


label
návěští
;
,
const
identifikátor
=
konstanta
;
type
identifikátor
=
typ
;
,
var
identifikátor
:
typ
;
;
identifikátor direktivy
blok
;
hlavička procedury
hlavička funkce
begin
příkaz
end
;

B PŘÍRUČKA

• literál

7653	0	958	–128	0256
85.6	–59.638	5E16	+18 .5E–1	+65.9E+3
'a'	'X'	'0'	'%'	''''
'TOTO JE RETEZEC'	' '	'vyjadreni apostrofu ''. '		

• identifikátor

I	j	EDITOR	*chr*	*text*	*write*
ToToJeIdEnTiFiKaToR		POM1	POM2	*integer*	Telefon
read	a24X2545				

• komentář

{ komentář – vysvětlující text }
(* komentář – vysvětlující text *)

• vyhrazená slova

and	**array**	**begin**	**case**	**const**	**div**
do	**downto**	**else**	**end**	**file**	**for**
function	**goto**	**if**	**in**	**label**	**mod**
nil	**not**	**of**	**or**	**packed**	
procedure	**program**	**record**	**repeat**	**set**	**then**
to	**type**	**until**	**var**	**while**	**with**

• výčtový typ

```
type
    DNY = (PONDELI, UTERY, STREDA, CTVRTEK, PATEK,
           SOBOTA, NEDELE);
var
    DEN: DNY;
```

```
DEN := PONDELI;

ord (PONDELI) = 0          ord (STREDA) = 2
ord (NEDELE) = 6           succ (STREDA) = CTVRTEK
pred (SOBOTA) = PATEK

DEN = PONDELI              DEN <> UTERY
PONDELI < NEDELE           SOBOTA > PONDELI
CTVRTEK <= PATEK           STREDA >= UTERY
```

• boolean

```
var
    LOGICKA1, LOGICKA2: boolean;

LOGICKA1 := true;
write (LOGICKA1);          writeln (LOGICKA2:7);

ord (true) = 1             ord (false) = 0
succ (false) = true        pred (true) = false
LOGICKA1 = true            LOGICKA1 <> false
false < true               true > false
false <= true              true >= false

LOGICKA1 and LOGICKA2      LOGICKA1 or LOGICKA2
not LOGICKA1
```

• integer

```
const
    maxint = maximální možné celé číslo;
var
    I, J, K: integer;

I := 0;                    read (J);
readln (K);                write (I:5);
readln (J,K);

ord (0) = 0                ord (-345) = -345
ord (78) = 78              succ (I) = I + 1
pred (J) = J-1             I = 0
I <> 8                     -345 < 78
```

```
78 > -345                    I < = 123
I > = 0

I + J                        I- J
I * J                        I div J
I mod J

abs (I)
```

• real

```
var
      A, B, C: real;

A := 10.05;                  read (B);
readln (A);                  write (A:10:5);
writeln (B:8,C);             write (C);

A = 10.05                    A <> - 45.7E - 3
- 45.7E-3 < 10.05            10.05 > - 45.7E - 3
A < = 34.8E2                 A > = 78.98E - 45

A + B                        A - B
A * B                        A / B

sqrt (A)                     sqr (A)
sin (A)                      cos (A)
arctan (A)                   ln (A)
exp (A)                      abs (A)
round (A)                    trunc (A)
```

• char

```
var
      C1, C2, C3: char;

C1 := 'A';                   read (C2);
readln (C3);                 write (C1:5);
writeln (C2:2,C3);

ord ('A') = 65               ord ('0') = 48
ord ('d') = 100              succ('B') = 'C'
```

• soubor

```
type
      SOU1 = file of integer;
      SOU2 = file of ZAZ1;

var
      S1 : SOU1;                    I: integer;
      S2 : SOU2;                    A: real;
      S3 : text;                    K: char;

S1^ := 576;                         get (S1);
put (S1);                           I := S1^;
reset (S1);                         rewrite (S1);

S2^.S2 := NEDELE;                   eof (S2)
read (S2,Z1);                       write (S2,Z1A);

read (S3,I);                        readln (S3,A);
write (S3,I:5,A:9);                 writeln (S3,K);
eoln (S3)                           page (S3);
```

• množina

```
type
      MNO1 = set of DNY;
      MNO2 = set of char;
      MNO3 = set of 0..127;

var
      M1A,
      M1   : MNO1;
      M2   : MNO2;
      M3   : MNO3;

M1 := M1A;                          M1A := [ ];
M1: = [PONDELI, UTERY];

M1 + M1A                            M1 * [PATEK..NEDELE]
M1A – [UTERY..PATEK]
M1 = M1A                            M1 < > [PONDELI]
M1A < = M1                          M1A > = M1
```

```
CTVRTEK in M1                    'a' in M2

M2 := [ ];                       M2:=['A'..'Z','0'..'9']

M3 := [I * 3,100]                M3 := [ ];
I in [1..10, 90..112]
```

• ukazatel

```
type
    UKA1 = ^UZEL;
    UZEL =
        record
            CO : integer;
            DALSI : UKA1;
        end; { UZEL }

var
    U1 : UKA1;
    U2 : UKA1;

new (U1);                        U1^.CO := 345;
new (U1^.DALSI);                 U2:=U1^.DALSI^.DALSI;
U1 = U2                          U1<>U2
U1 := U2;                        dispose (U2);
```

• priorita operátorů

1.	=	<>	<	>	<=	>=	**in**
2.	+	–	**or**				
3.	*	/	**div**	**mod**	**and**		
4.	**not**						

• příkazy

proměnná := *výraz*	přiřazení
identifikátor(*výraz*, ... , *výraz*)	příkaz procedury
goto *návěští*	skok na návěští

prázdný příkaz

složený příkaz

```
begin
        příkaz;
        :
        příkaz
end
```

neúplný podmíněný příkaz

```
if výraz
        then
                příkaz
```

úplný podmíněný příkaz

```
if výraz
        then
                příkaz
        else
                příkaz
```

příkaz case

```
case výraz of
        konstanta, ..., konstanta : příkaz;
        :
        konstanta, ... , konstanta : příkaz;
end
```

cyklus for

```
for proměnná := výraz to výraz do
        příkaz

for proměnná := výraz downto výraz do
        příkaz
```

cyklus while

```
while výraz do
        příkaz
```

cyklus repeat

```
repeat
        příkaz;
        :
        příkaz
until výraz
```

příkaz with

```
with proměnná,... , proměnná do
                příkaz
```

Tabulka znakového kódu ASCII

znak	klávesa	*ord*(znak)	osmičkově	šestnáctkově
NUL	Ctrl @	0	000	00
SOH	Ctrl A	1	001	01
STX	Ctrl B	2	002	02
ETX	Ctrl C	3	003	03
EOT	Ctrl D	4	004	04
ENQ	Ctrl E	5	005	05
ACK	Ctrl F	6	006	06
BEL	Ctrl G	7	007	07
BS	Ctrl H	8	010	08
TAB	Ctrl I	9	011	09
LF	Ctrl J	10	012	0A
VT	Ctrl K	11	013	0B
FF	Ctrl L	12	014	0C
CR	Ctrl M	13	015	0D
SO	Ctrl N	14	016	0E
SI	Ctrl O	15	017	0F
DLE	Ctrl P	16	020	10
DC1	Ctrl Q	17	021	11
DC2	Ctrl R	18	022	12
DC3	Ctrl S	19	023	13
DC4	Ctrl T	20	024	14
NAK	Ctrl U	21	025	15
SYN	Ctrl V	22	026	16
ETB	Ctrl W	23	027	17
CAN	Ctrl X	24	030	18
EM	Ctrl Y	25	031	19
SUB	Ctrl Z	26	032	1A
ESC	Ctrl [	27	033	1B
FS	Ctrl \	28	034	1C
GS	Ctrl]	29	035	1D
RS	Ctrl ^	30	036	1E
US	Ctrl _	31	037	1F
' '	mezerník	32	040	20
'!'	!	33	041	21
'"'	"	34	042	22
'#'	#	35	043	23
'$'	$	36	044	24
'%'	%	37	045	25
'&'	&	38	046	26
'''	'	39	047	27

znak	klávesa	*ord*(znak)	osmičkově	šestnáctkově
'('	(	40	050	28
')'	)	41	051	29
'*'	*	42	052	2A
'+'	+	43	053	2B
','	,	44	054	2C
'-'	-	45	055	2D
'.'	.	46	056	2E
'/'	/	47	057	2F
'0'	0	48	060	30
'1'	1	49	061	31
'2'	2	50	062	32
'3'	3	51	063	33
'4'	4	52	064	34
'5'	5	53	065	35
'6'	6	54	066	36
'7'	7	55	067	37
'8'	8	56	070	38
'9'	9	57	071	39
':'	:	58	072	3A
';'	;	59	073	3B
'<'	<	60	074	3C
'='	=	61	075	3D
'>'	>	62	076	3E
'?'	?	63	077	3F
'@'	@	64	100	40
'A'	A	65	101	41
'B'	B	66	102	42
'C'	C	67	103	43
'D'	D	68	104	44
'E'	E	69	105	45
'F'	F	70	106	46
'G'	G	71	107	47
'H'	H	72	110	48
'I'	I	73	111	49
'J'	J	74	112	4A
'K'	K	75	113	4B
'L'	L	76	114	4C
'M'	M	77	115	4D
'N'	N	78	116	4E
'O'	O	79	117	4F

znak	klávesa	*ord*(znak)	osmičkově	šestnáctkově
'P'	P	80	120	50
'Q'	Q	81	121	51
'R'	R	82	122	52
'S'	S	83	123	53
'T'	T	84	124	54
'U'	U	85	125	55
'V'	V	86	126	56
'W'	W	87	127	57
'X'	X	88	130	58
'Y'	Y	89	131	59
'Z'	Z	90	132	5A
'['	[	91	133	5B
'\'	\	92	134	5C
']'	]	93	135	5D
'^'	^	94	136	5E
'_'	_	95	137	5F
'`'	`	96	140	60
'a'	a	97	141	61
'b'	b	98	142	62
'c'	c	99	143	63
'd'	d	100	144	64
'e'	e	101	145	65
'f'	f	102	146	66
'g'	g	103	147	67
'h'	h	104	150	68
'i'	i	105	151	69
'j'	j	106	152	6A
'k'	k	107	153	6B
'l'	l	108	154	6C
'm'	m	109	155	6D
'n'	n	110	156	6E
'o'	o	111	157	6F
'p'	p	112	160	70
'q'	q	113	161	71
'r'	r	114	162	72
's'	s	115	163	73
't'	t	116	164	74
'u'	u	117	165	75
'v'	v	118	166	76
'w'	w	119	167	77

znak	klávesa	*ord*(znak)	osmičkově	šestnáctkově
'x'	x	120	170	78
'y'	y	121	171	79
'z'	z	122	172	7A
'{'	{	123	173	7B
'|'	|	124	174	7C
'}'	}	125	175	7D
'~'	~ (tilda)	126	176	7E
del	del	127	177	7F

Znaky s ordinálními čísly 0 až 31 jsou řídicími znaky. Na klávesnici se zapisují současným stlačením klávesy **Ctrl** a některé další klávesy. Na obrazovce nebo jiném výstupním zařízení jsou obvykle nezobrazitelné. Řídicí znaky jsou pojmenovány zkratkami vycházejícími z jejich anglických názvů.

Nejpoužívanější jsou následující řídicí znaky:

- BEL – zvukový signál,
- BS – posun o znak zpět,
- TAB – tabelátor,
- LF – posun na další řádek,
- VT – vertikální tabelátor,
- FF – posun na začátek další stránky,
- CR – návrat vozíku na začátek řádku.

LITERATURA

[1] ATKINSON,L.V.: A Student's guide to Programming in Pascal, London, John Wiley a Sons 1982

[2] ATKINSON,L.V.: Pascal Programming. London, John Wiley a Sons 1980

[3] BOWEN,K.A.: Speaking Pascal. New Jersey, Hayden Book Comp.1981

[4] BUTOMO,I.D. – SAMOČADIN,A.V. – USANOVA,D.V.: Programirovanie na algoritmičeskom jazyke Pascal dlja mikro EVM. Leningrad, Izdatelstvo leningradskovo universiteta 1985

[5] COLLINS,W.J.: Intermediate Pascal Programming. New York, McGraw – Hill Book Company 1986

[6] COLLINS,W.J.: An Introduction to Programming in Pascal. New York, Macmillan Publishing Company 1984

[7] EISENBACH,S. – SADLER,C.: Pascal for Programmers. Berlin, Springer Verlag 1981

[8] FAUSTUS,L. – POLÍVKA,F.: Botanický klíč. Praha, SPN 1975

[9] GROGONO,P.: Programming in Pascal. Reading, Adison – Wesley 1980

[10] HAMZA,J. a kol.: Rok s Evou. Bratislava, Obzor 1984

[11] HORÁK,Z. – KRUPKA,F.: Fyzika. Praha, SNTL 1976

[12] ISO/TC97/SC5 Programming Languages Specification for Computer; Programming Language Pascal. Third Draft Proposal ISO/DP7185, 1981

[13] JENSEN,K. – WIRTH,N.: Pascal – rukovodstvo dlja polzovatela i opisanie jazyka. Moskva, Finansy i statistika 1982 (překlad z angličtiny)

[14] JINOCH,J. – MÜLLER,K. – VOGEL,J.: Programování v jazyku Pascal. Praha, SNTL 1987

[15] KERNIGHAN,B.W.: Software Tools in Pascal. New York, Addison – Wesley Publishing Company 1981

[16] KIEBURTZ,L.: Structured Programming and Problem Solving with Pascal. London, Prentice Hall 1977

[17] KOHLMAN,Č.: Matematika sdělovací techniky. Praha, Technicko – vědecké vydavatelství 1951

[18] LEWIS,T.: Pascal Programming for the Apple. London, Prentice Hall 1981

[19] MOLNÁR,L.: Programovanie v jazyku Pascal. Bratislava, Alfa 1987

[20] OCHRANOVÁ,R.: Úvod do programování. [Skriptum.] Brno, UJEP 1982

[21] PAULIN,G. – SCHIEMANGK,H.: Programmieren mit Pascal. Berlin, Akademie Verlag 1981

[22] PERMINOV,O.N.: Jazyk programirovanija Pascal. Moskva, Radio i svjaz 1983

[23] RÁBOVÁ,Z. a kol.: Počítače a programování. [Skriptum.] Brno, VUT 1980

[24] RŮŽIČKA,J. – HRUŠKA,E.: Fyzika. [Skriptum.] Brno, VŠZ 1983

[25] SCHNEIDER,G.M. – WEINGART,S.W. – PERELMANN,D.M.: An Introduction to Programming and Problem Solving in Pascal. New York, John Wiley & Sons 1978

[26] Turbo Pascal 4.0, manuál firmy Borland International 1987

[27] WIRTH,N.: Algoritmy + struktury dannych = programy. Moskva, Mir 1985 (překlad z angličtiny)

[28] WIRTH,N.: Systematické programovanie. Bratislava, Alfa 1981

[29] WIRTH,N.: The Programming Language Pascal. Acta Informatica, 1, 1971, str. 35 – 63

REJSTŘÍK

Ing. Tomáš Hruška, CSc.

PASCAL PRO ZAČÁTEČNÍKY

DT 800.92
681.3.04

Vydalo SNTL — Nakladatelství technické literatury, n. p.,
Spálená 51, 113 02 Praha 1
v roce 1989
jako svou 11 038. publikaci
Redakce teoretické literatury
Odpovědná redaktorka RNDr. Irena Bergerová
Vazbu navrhl Vladislav Jacák
Grafická úprava Ing. Tomáš Hruška, CSc.,
Technická redaktorka Viola Urbanová
Vytiskly Tiskařské závody, s. p., závod 2, Praha 1,
Karmelitská 6
368 stran, 38 obrázků, 14 ilustrací, 3 tabulky
Typové číslo L11-E1-V-85/12087 Vydání první
Náklad 35 000 výtisků. 18,55 AA, 19,19 VA

03/2
Cena vázaného výtisku Kčs 40,—
505/21, 856

Publikace je určena nejširší veřejnosti, tj. všem, kteří mají zájem se naučit programovat v jazyku Pascal.

04-003-89 Kčs 40,—