O B S A H :

		Strana:
Kapitola 1	Předmluva	5
1/1	Charakteristika této příručky	7
1/2	Překlad a běh programu	8
1/3	Síla zápisu	9
Kapitola 2	Syntaxe a semantika	11
2/1	Identifikátory	11
2/2	Celáčísla bez znaménka	11
2/3	Čísla bez znaménka	11
2/4	Konstanta bez znaménka	12
2/5	Konstanta	13
2/6	Jednoduchý typ	13
2/7	Typy	14
2/7/1	Pole a množiny	14
2/7/2	Ukazatele	15
2/7/3	Záznamy	16
2/7/4	Seznam položek	16
2/9	Proměnné	17
2/1Ø	Faktor (sčítanec)	18
2/11	TERM (činitel)	18a
2/12	Jednoduché výrazy	18
2/13	Výrazy	18
2/14	Seznam parametrů	19
2/15	Příkazy	19
2/16	Programový blok	21
2/17	Program	22

		Strana:
Kapitola 3	Předdefinované identifikátory	23
	3/1 Konstanty	23
	3/2 Typy	23
	3/3 Procedury a funkce	23
	3/3/1 Procedury vstupu a výstupu	23
	3/3/1/1 WRITE	23
	3/3/1/2 WRITELN	26
	3/3/1/3 PAGE (stránka)	26
	3/3/1/4 READ (čtení)	27
	3/3/1/5 READLN	29
	3/3/2 Vstupní funkce	29
	3/3/2/1 EOLN	29
	3/3/2/2 INCH	29
	3/3/3 Funkce převodů	29
	3/3/3/1 TRUNC (X)	29
	3/3/3/2 ROUND (X)	30
	3/3/3/3 ENTIER (X)	30
	3/3/3/4 ORD (X)	30
	3/3/3/5 CHR (X)	30
	3/3/4 Aritmetické funkce	31
	3/3/4/1 ABS (x)	31
	3/3/4/2 SQR (X)	31
	3/3/4/3 SQRT (X)	31
	3/3/4/4 FRAC (X)	31
	3/3/4/5 SIN (X)	31
	3/3/4/6 COS (X)	31
	3/3/4/7 TAN (X)	32
	3/3/4/8 ARCTAN (X)	32
	3/3/4/9 EXP (X)	32
	3/3/4/10 LN (X)	32

		Strana:
	3/3/5 Další předdefinované procedury	32
	3/3/5/1 NEW (P)	32
	3/3/5/2 MARK (v1)	32
	3/3/5/3 RELEASE (v1)	33
	3/3/5/4 INLINE (c1,c2,c3,...)	33
	3/3/5/5 USER (v)	33
	3/3/5/6 HALT	34
	3/3/5/7 POKE (x,v)	34
	3/3/5/8 TOUT (NAME,START,SIZE)	34
	3/3/5/9 TIN (NAME,SIZE)	35
	3/3/5/10 OUT (P,C)	35
	3/3/6 Další předdefinované funkce	36
	3/3/6/1 RANDOM	36
	3/3/6/2 SUCC (X)	36
	3/3/6/3 PRED (X)	36
	3/3/6/4 ODD (X)	36
	3/3/6/5 ADDR (V)	36
	3/3/6/6 PEEK (X,T)	37
	3/3/6/7 SIZE (V)	37
	3/3/6/8 INP (P)	37
Kapitola 4	Poznámky a varianty kompilátoru	38
	4/1 Poznámky	38
	4/2 Varianty kompilátoru	38
Kapitola 5	Integrální editor	42
	5/1 Úvod do editoru	42
	5/2 Povely editoru	43
	5/2/1 Vkládání textu	43
	5/2/2 Vypisování textu	44
	5/2/3 Úpravy textu	45

		Strana:
	5/2/4 Povely pro magnetofon	49
	5/2/5 Překlad a spuštění z editoru	5Ø
	5/2/6 Ostatní povely	51
	5/3 Příklad použití editoru	52
Doplněk 1	Chyby	54
	A/1/1 Chyby generované kompilátorem	54
	A/1/2 Chybové zprávy při běhu programu	56
Doplněk 2	Rezervované slova a předdefinované identifikátory	58
	A/2/1 Rezervované slova	58
	A/2/2 Speciální symboly	58
	A/2/3 Předdefinované identifikátory	58
Doplněk 3	Vnitřní zobrazení a ukládání dat	6Ø
	A/3/1 Vnitřní zobrazení dat	6Ø
	A/3/1/1 Celá čísla	6Ø
	A/3/1/2 Znaky, booleanské výrazy a ost.skaláry	6Ø
	A/3/1/3 Reálná čísla	61
	A/3/1/4 Zápisy a pole	62
	A/3/1/5 Množiny	63
	A/3/1/6 Ukazatele	63
	A/3/2 Ukládání proměnných za běhu programu	63
Doplněk 4	Příklady programů HP4T	66
Literatura		71
Aplikace	Zvuk a grafika pro HP4T na ZX SPEKTRU	72

K A P I T O L A 1

P Ř E D M L U V A

HISOFT PASCAL (HP 4T) je rychlá, obsažná a snadno použitelná verze jazyka PASCAL, tak jak je definován v knize "Pascal User Manual and Report"(Jensen - With).

Vyjímky oproti této specifikaci jsou následující :

Soubory nejsou implementovány (zabudovány)

Záznamy nemohou mít variantní část

Procedury a funkce nemohou být parametry

Mnoho procedur a funkcí je předefinováno, aby reflektovaly na prostředí,ve kterém je mokropočítač použit. Například POKE? PEEK, TIN, ADDR!

Vlastní překladač potřebuje asi 12KB.

Podpůrné programy nutné pro běh přeloženého kódu (runtimes) potřebují asi 4KB. Toto vše je připraveno na pásce. Spojení mezi HP4T a počítačem je uskutečněno pomocí vektorů, konvenčně umístěných na začátku podpůrných programů.

Kdykoliv HP 4 T očekává vstup řádky, je význam řídících znaků následující.

ENTER (v textu jako RETURN) ukončení řádky

EDIT (v textu jako CC) návrat do editoru)

DELETE (v textu jako CH) vymaž poslední znak

→ (v textu jako CI) posun na následující TAB pozici;

← (v textu jako CK) vymaž celou řádku

(v textu jako CP) směruje výstup na tiskárnu (je-li připojena) nebo byl-li přímý výstup na tiskárnu návrat na obrazovku.

Když nahrajete překladač do paměti (LOAD ""), program se rozběhne a položí vám otázku :

Top of RAM?

Můžete zadat celé číslo až do 65 536 (nejvyšší adresa, která je překladači k dispozici) plus jedna. Nebo pouze ENTER a počítač najde nejvyšší adresu sám. (V tomto případě umístí vrchol paměti pod oblast uživatelské grafiky UDG).

Další otázka bude :

Top of RAM for "T"

Můžete vložit číslo nebo jen ENTER. Toto číslo označuje vrchol paměti pro programy přeložené povelem T. Pokud stisknete jenom ENTER, použije se stejná hodnota jako byla zadána pro Top of RAM.

Nakonec se objeví :

Table size?

To co nyní vložíte určuje rozsah paměti určené pro tabulku symbolů překladače. Opět můžete vložit číslo následované ENTER nebo pouze ENTER. V druhém případě vyhradí počítač tabulku symbolů o kapacitě 1/16 dostupné paměti.

Tabulka symbolů nesmí přesáhnout adresu 32 768 dec. Jestliže zadáte větší hodnotu, pak se otázka Top of RAM? a další budou opakovat.

Můžete také před číslo v poslední otázce přidat znak 'É ' - pokud tak učiníte nebude interní řádkový editor udržován v paměti a tak máte možnost připojit svůj vlastní editor.

Nyní bude překladač a řádkový editor (pokud má zůstat v paměti) přemístěn za tabulku symbolů a řízení bude předáno editoru.

Některé další podrobnosti lze nalézt v "HP4T Alterna-

tion Guide".

Poznámka : čísla, které předchází znak # jsou čísla hexadecimální.

1/1 Charakteristika této příručky.

Tato příručka nemá za účel učit PASCAL, k tomu jsou některé knihy uvedené v seznamu literatury, objasňující programování v PASCALu.

Příručka je referenční dokument popisující zvláštnosti HISOFT PASCAL 4T.

Kapitola 2. říká jakou syntaxi a obsah očekává kompilátor.

Kapitola 3. probírá různé předdefinované identifikátory obsažené v HP4T, od konstant po funkce.

Kapitola 4. obsahuje informace o různých dostupných možnostech kompilace a také o formátu komentářů.

Kapitola 5. ukazuje jak použít řádkový editor, který je nedílnou součástí HP4T (jestliže nechcete používat tento editor, ale Váš vlastní nebo jiný, tak najdete radu v "HP4T Alternation Guide".

Výše uvedené části čtěte pečlivě, pomůže Vám to při používání HP4T.

Doplněk 1. upřesňuje zprávy o chybách při překladu a běhu.

Doplněk 2. je výpis rezervovaných slov a vymezených identifikátorů.

Doplněk 3. podává podrobnosti o vnitřním zobrazení dat v HP4T - užitečné pro programátory, kteří si chtějí uěpinit ruce.

Doplněk 4. uvádí příklady několika programů v

PASCALU - prostudujte je, získáte zkušenosti při řešení problémů v HP4T.

1/2 Překlad a běh programu.

Podrobnosti jak tvořit, zlepšovat a spouštět HP4T program s použitím řádkového editoru jsou v kapitole 4. Informace co se stane, když použijete vlastní editor - "HP4T Alternation Guide".

Kompilátor generuje výpis v následujícím tvaru :

xxxx nnnn text zdrojové řádky

kde xxxx je adresa na niž začíná kód generovaný touto řádkou

nnnn je číslo řádky s potlačenými nulami před číslem

Jestliže řádka obsahuje více než 32 znaků, tak kompilátor vloží novou řádku. Takto řádka nemůže být nikdy delší než 80 znaků.

Chcete-li, může být výpis směrován do tiskárny pomocí varianty P (viz. kapitola 4.).

Výpis můžete kdykoliv zastavit stisknutím SPACE; potom použitím EDIT se vrátíte zpět do editoru a stiskem některé jiné klávesy opět spustíte výpis.

Jestliže je během překladu zjištěna chyba, zobrazí se zpráva * ERROR * následovaná šipkou vzhůru ("^"), která ukazuje na symbol následující po symbolu, který chybu způsobil a číslem chyby (viz doplněk 1.). Výpis se přeruší; stiskem "E" se editor vrátí a edituje zobrazovanou řádku, "P" vrátí editor a edituje předcházející řádku (když existuje), nebo kteroukoliv jinou klávesou pokračuje kompilace.

Jestliže je program nesprávně ukončen (např. chybí END), tak následuje zpráva "No more text" (není další text) na displeji a řízení se vrátí editoru.

Dostane-li se kompilátor mimo rozsah tabulky, zobrazí se zpráva "No Table Space" a řízení se opět vrátí do editoru. Normálně může programátor přehrát program na pásku, znovu nahrát kompilátor a specifikovat větší "Table size"; viz kapitolu 1.

Pokud překlad skončí správně, ale v textu byly chyby, potom se zobrazí počet chyb a řízení se vrátí editoru přičemž se cílový kód vymaže. V případě, že je bezchybný, tak se zobrazí zpráva "Run?"; chcete-li okamžitě spustit program, tak odpovězte "Y", jinak se řízení vrátí do editoru.

Během chodu cílového programu(kódu) se mohou generovat různé zprávy o programových chybách, viz doplněk 1. Běh můžete zastavit "SPACE"; následným použitím "EDIT" přerušíte běh nebo libovolnou jinou klávesou jej znovu spustíte.

1/3 Síla zápisu

Jednotlivé jazyky zajišťují rozdílným způsobem, aby uživatel nepoužil některý datový prvek způsobem, který se neslučuje s jeho definicí.

Nejnižší úroveň má strojový kód, kde není žádný způsob kontroly. Následuje BYTE "Tiny Pascal" kde znaky, celá čísla a logické proměnné mohou být smíchány bez hlášení chyby. O něco výše je BASIC, který rozezná čísla a řetězce, někdy mezi zápisy celých a reálných čísel. PASCAL přichází až k rozlišování nečíselných znaků a vrcholem jsou jazyky jako ADA, které přesně rozlišují neslučitelné zápisy čísel.

V zásadě existují dva přístupy, kterých PASCAL, při realizaci užívá, aby zvýraznil zápis strukturální ekvivalence, nebo jmenná ekvivalence HP4T využívé jmenné ekvivalence pro RECOREy a ARRAYc. Jaké to má důsledky vysvětluje kapito-

tola 2. - vysvětleme si to zde na příkladu; řekněme, že dvě proměnné jsou definované následujícím způsobem :

```
VAR A : ARRAY ['A'..'C'] OF INTEGER;
    B : ARRAY ['A'..'C'] OF INTEGER;
```

První nás může svádět k jednoduchému zápisu A:=B ale toto generuje chybu (* ERROR * 1Ø) v HP4T, protože nahoře uvedenou definicí byly dva separátní zápisy (Type recordy). Jinými slovy uživatel neuvažoval, že A a B měli reprezentovat stejný typ dat. Měl udělat toto :

```
VAR A, B : ARRAY ['A'.. 'C'] OF INTEGER;
```

A nyní může uživatel podle své vůle přiřadit A k B a naopak, neboť byl vytvořen jen jediný TYPE record.

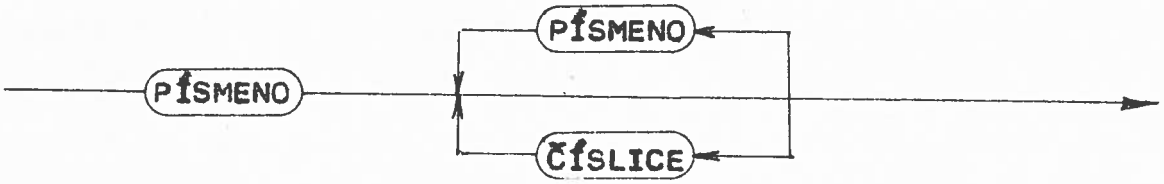
Ačkoliv při povrchním pohledu se může zdát tento přístup jmenných ekvivalentů poněkud komplikovaný, tak všeobecně vede k menšímu počtu programátorových chyb, neboť vyžaduje od programátora více přemýšlení.

K A P I T O L A 2

S Y N T A X E A S E M A N T I K A

Tato kapitola rozebírá syntaxi a semantiku HP4T, pouze pokud je odlišná od "Pascal User Manual and Report" (Jenson - Wirth).

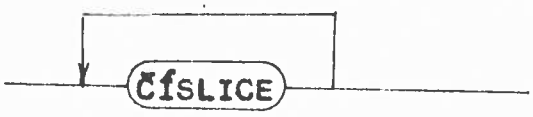
2/1 Identifikátory



Pouze deset prvních znaků je zpracováno jako významné.

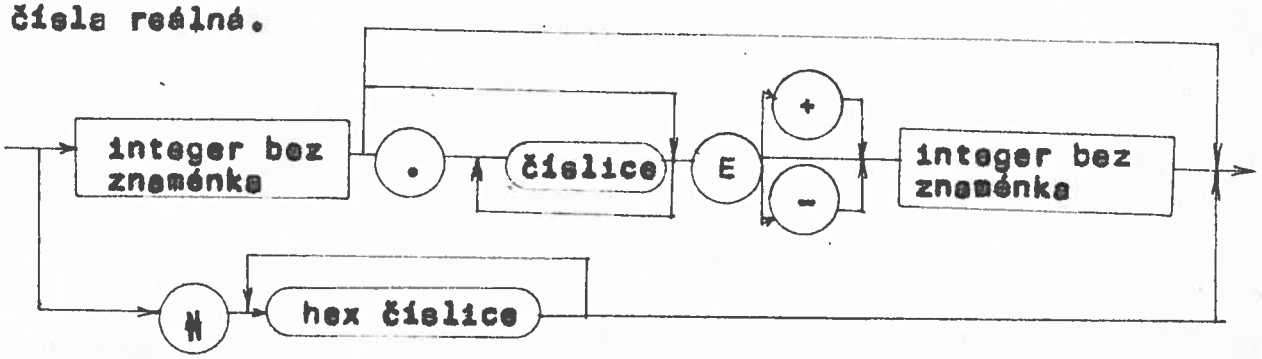
Identifikátory mohou obsahovat malé a velké písmena, přitom velká písmena nejsou převáděna na malá, takže identifikátory HELLO, HELlo a hello nejsou stejné. Rezervovaná slova a vymezené identifikátory musí být psány jen velkými písmeny.

2/2 Celá čísla bez znaménka



2/3 Čísla bez znaménka

V HP4T má celé číslo (integer) absolutní hodnotu menší nebo rovnu 32 767. Větší celé číslo jsou zpracovávána jako čísla reálná.



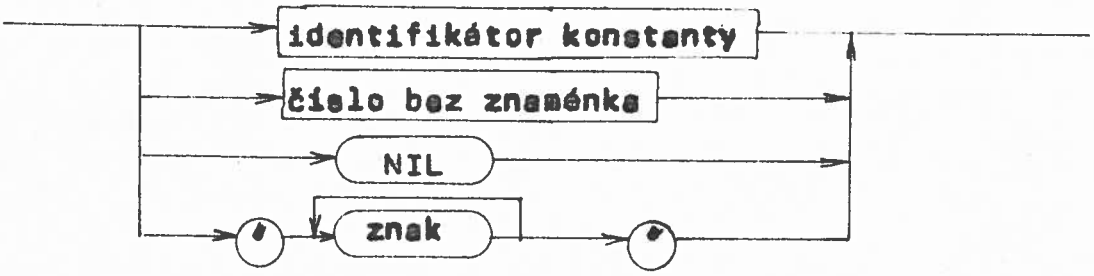
Mantisa reálného čísla má délku 23 bitů. Přesnost při použití reálného čísla je okolo 7 významných číslic. Všimněte si, že přesnost se ztratí, jestliže výsledek výpočtu je mnohem menší než absolutní hodnota argumentu, např. 2.ØØØØ2 - 2 nedá Ø.ØØØØ2. Nepřesnost je dána zobrazením desetinné části jako binární. Ale výsledek pro celá čísla střední délky je přesný, např. 2ØØØØ2 - 2ØØØØØ je přesně 2.

Reálná čísla mohou být v rozsahu 5.9E-39 až 3.4E38. Je zbytečné používat více než sedm čísel mantisy při specifikaci reálného čísla, protože nadbytečné číslice jsou ignorovány.

Je-li důležitá přesnost převedeme číslo do exponenciálního tvaru, takže například Ø.ØØØ123456 je zcela přesně reprezentováno 1.23456E-4.

Hexadecimální číslo specifikuje programátorovi adresu paměti při využívání assembleru. Všimněte si, že před hexadecimálním číslem musí být znak $ jinak je hlášena chyba (% ERROR % 5 :)

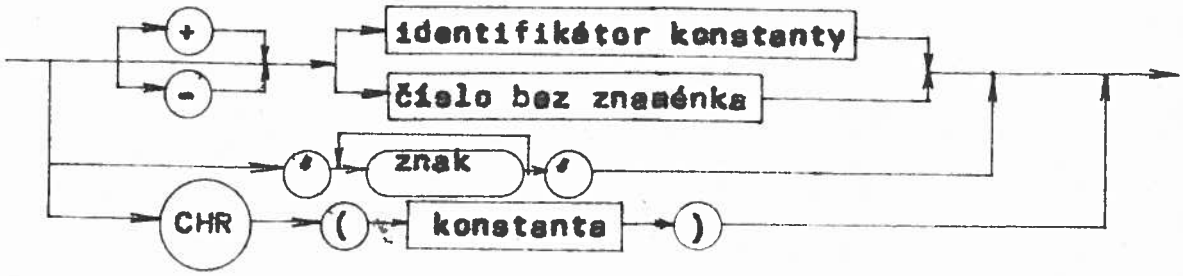
2/4 Konstanta bez znaménka



Všimněte si, že řetězec nesmí obsahovat více než 255 znaků. Řetězcové typy jsou ARRAY [1..N] OF CHAR, kde je celé číslo od 1 do 255, včetně. Řetězec písmen nesmí obsahovat znak(CHR (13)) (začátek-konec řádky) jinak je hlášena chyba (% ERROR % 68).

Znaky lze rozšířit na všech 256 prvků ASCII, což je plně kompatabilní se standartem Pascalu, nulový znak není reprezentován "; místo toho použijem CHR (ø).

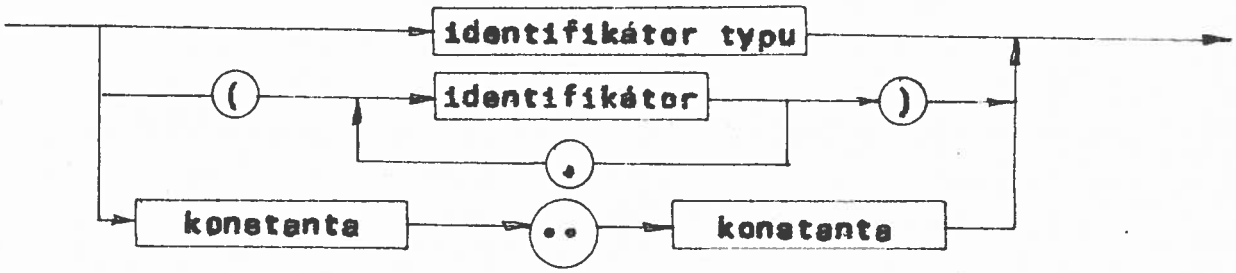
2/5 Konstanta



Nestandartně je do konstanty přiřazen CHR (znak) aby mohly být použity řídicí znaky. V tomto případě musí být konstanta v závorce celé číslo. Například :

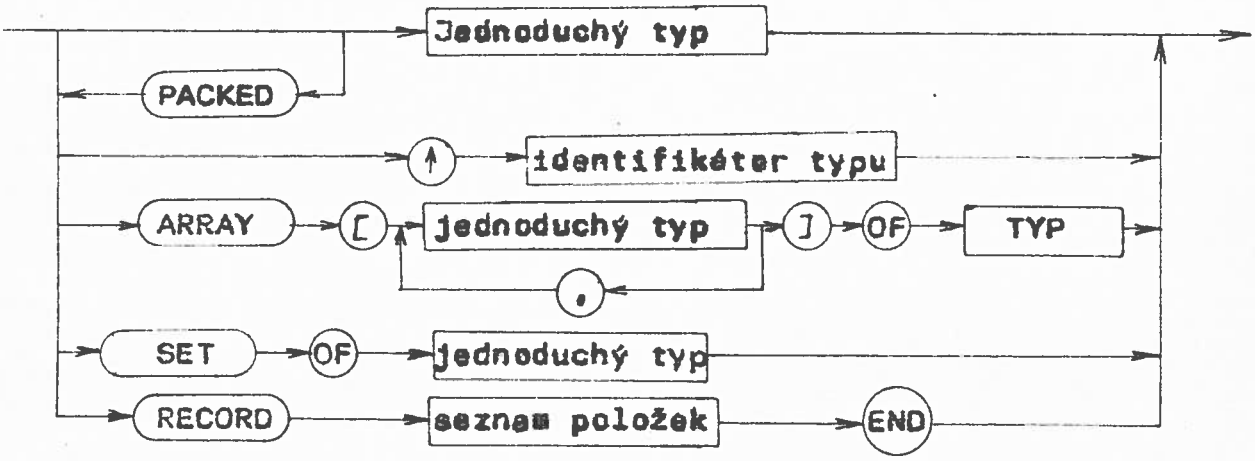
```
CONST bc= CHR (1ø);
      cr= CHR (13);
```

2/6 Jednoduchý typ



Výčtový typ (identifikátor, identifikátor, ...) nemůže mít více než 256 prvků.

2/7 Typy



Rezervované slovo PACKED je akceptováno, ale ignorováno neboť pole znaků a pod. jsou vždy v zhuštěném tvaru.

2

(SET), pokud vyžaduje zhuštění.

2/7/1 Pole a množiny

Základní typ množiny může mít nejvýše 256 prvků. To dovoluje deklarovat SET OF CHAR a SET sestávající z libovolného výčtového typu. Všimněte si, že pouze interval z celých čísel může být použit jako základní typ. Všechny podmnožiny celých čísel jsou zpracovávány jako SET OF Ø..255.

Je možné deklarovat pole polí, pole množin, záznam množin atd.

Dva zápisy se zpracovávají jako ekvivalentní, pouze když to pramení z jejich společné definice, společným rezervovaným slovem ARRAY. Proto následující zápisy nejsou ekvivalentní :

```
tablea = ARRAY [1..1ØØ] OF INTEGER;
tableb = ARRAY [1..1ØØ] OF INTEGER
```

Tedy proměnnou tablea nelze přiřadit proměnné tableb.

Toto umožňuje zjistit chybu, jako přiřazení dvou tabulek reprezentující rozdílná data. Výše uvedené omezení se nevztahuje na řetězce (ARRAY [1..n] OF CHAR), které jsou vždy považovány za ekvivalentní, pokud mají stejnou délku.

2/7/2 Ukazatele

HP4T dovoluje tvorbu dynamických proměnných s použitím standartní procedury NEW (viz kapitola 3). Dynamické proměnné na rozdíl od statických proměnných, které mají přidělen prostor v paměti náležející bloku, ve kterém jsou deklarovány, nemohou být určeny přímo identifikátory, protože je nemají, místo toho používají proměnné typu ukazatel.

Proměnná typu ukazatel, která je statickou proměnnou, obsahuje adresu dynamické proměnné a dynamická proměnná je sama o sobě přístupná po vložení znaku "↑" za proměnnou typu ukazatel. Příklady použití ukazatelů můžete studovat v doplňku 4.

V HISOFT PASCALu 4 jsou některá omezení :

Ukazatele na dosud nedefinované typy nejsou dovoleny. To svým nebrání konstrukci dynamického seznamu, neboť definice typu může obsahovat ukazatele na sebe sama. Například : TYPE

```
item = RECORD
         value : INTEGER;
         next : ↑item
         END;
link = ↑ item;
```

Ukazatel na ukazatel není dovolen.

Ukazatele na stejné typy jsou považovány za ekvivalentní. Například :

```
VAR
    first : link;
    current : ↑item
```

Proměnná first a current jsou stejného typu (to jest používá se ekvivalence podle struktury) a můžeme je navzájem přiřazovat nebo porovnávat.

Je předefinována konstanta NIL.

Pokud je NIL přiřazena proměnné typu ukazatel, potom proměnná neobsahuje žádnou adresu.

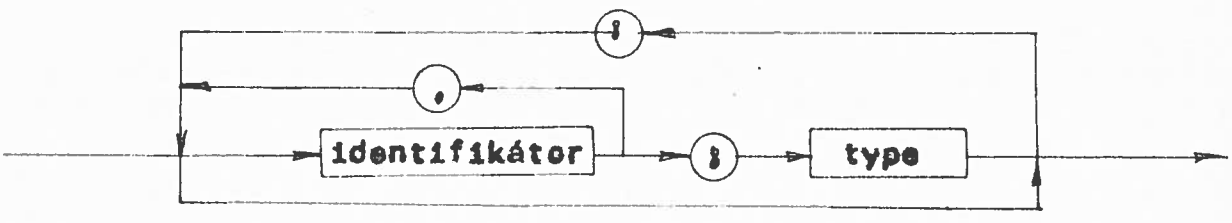
2/7/3 Záznamy

Implementace typu RECORD, struktury pozůstávající z pevně daného počtu pojmenovaných položek, je v HP4T stejná jako ve standartu PASCALu, s výjimkou, že nejsou dovoleny varientní záznamy.

Dva záznamy jsou zpracovávány jako stejné v případě, že jsou definovány jedním rezervovaným slovem RECORD (viz předchozí odstavec 2/7/1).

Příkaz WITH můžeme použít k přístupu rozdílných polí uvnitř zápisů v kompaktnější formě. Viz doplněk 4. kde je příklad obvyklého použití WITH a RECORD.

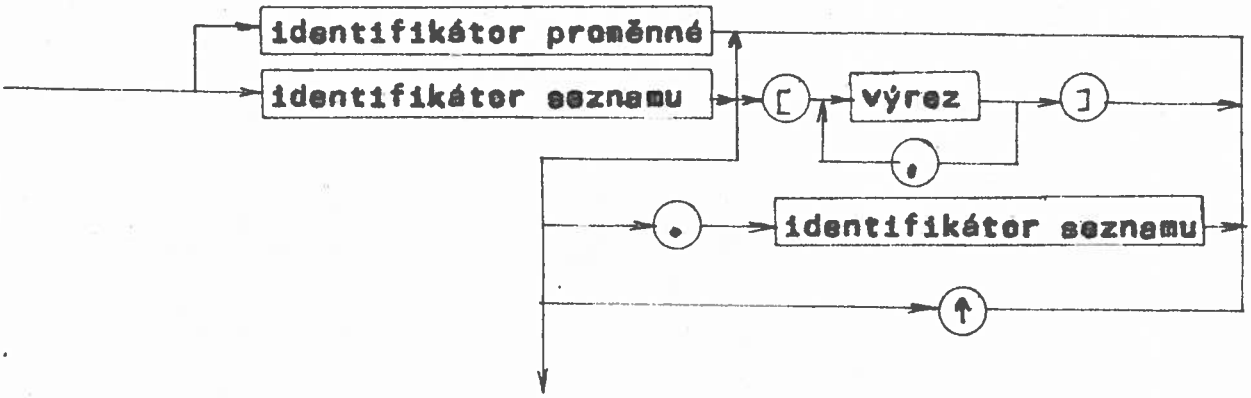
2/8 Seznam položek



Užívá se ve spojení s RECORD, viz předcházející odsta-

vec 2/7/3 nebo doplněk 4..

2/9 Proměnné



HISOFT PASCAL 4 má dva druhy proměnných , statické a dynamické. Statické proměnné jednoznačně deklarujeme pomocí VAR, v paměti se vymezí pro ně místo v bloku, kde jsou deklarovány.

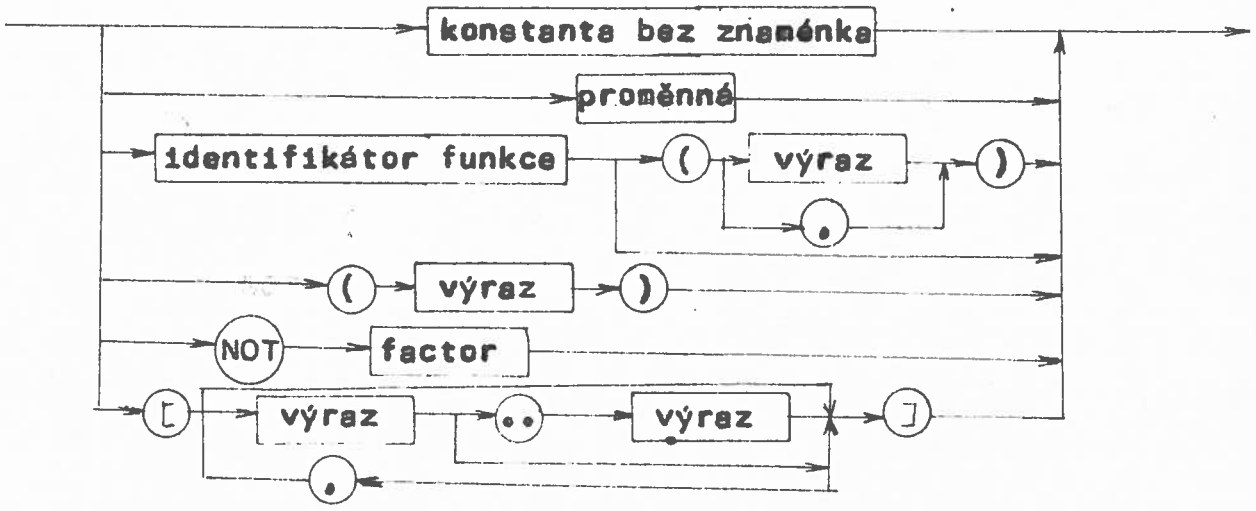
Dynamickou proměnnou vytvoříme během programu procedurou NEW. Nejsou předem deklarovány a nejsou přístupné pomocí identifikátorů, odkaz je nepřímý přes statickou proměnnou typu ukazatel, která obsahuje adresu dynamické proměnné. Viz kapitola 2/7/2 a 3. Více podrobností a příklady použití jsou v dodatku 4..

Když programátor definuje prvky z vícerozměrného pole, není nucen při odkazu na proměnnou použít stejný formát indexu jako při deklaraci. To je změna proti HISOFT PASCAL 3. Například je-li proměnná deklarována :

ARRAY [1..10] OF ARRAY [1..10] OF INTEGER

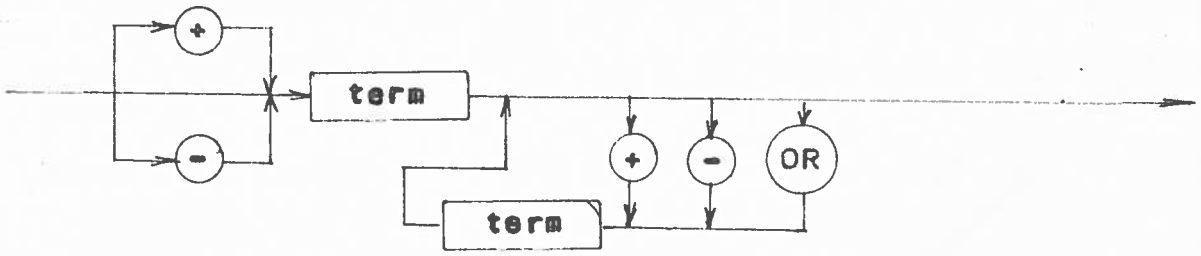
potom [1][1] nebo [1,1] může být použito pro volání prvku pole (1,1).

2/10 Faktor (sčítanec)



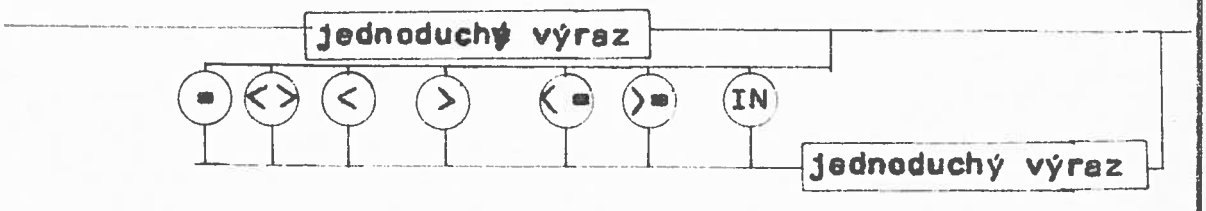
Viz výrazy v kapitole 2/13 a funkce v kapitole 4 kde je více podrobností.

2/12 Jednoduchý výraz



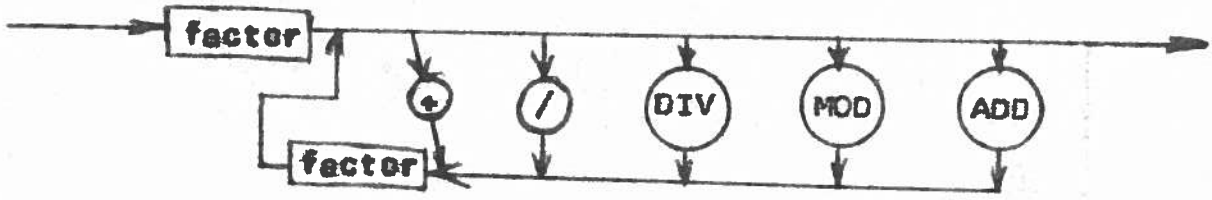
Komentář týkající se jednoduchých výrazů najdete v kapitole 2/11.

2/13 Výrazy



Při použití relace IN může mít levý operand libovolnou

2/11 TERM (činitel)

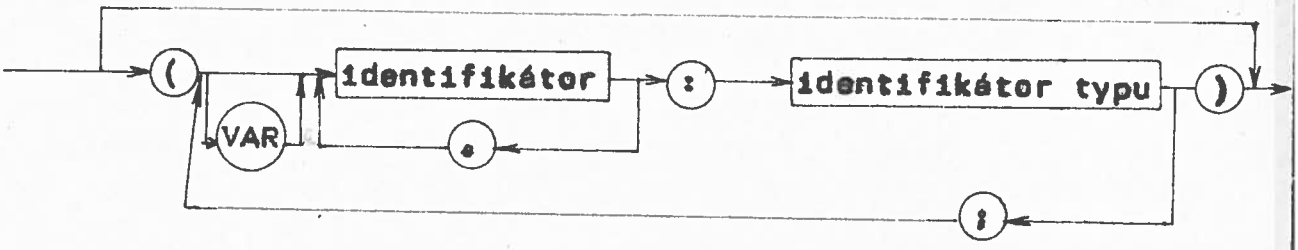


Dolní hranice množiny je vždy nula. Velikost množiny je vždy maximum základní množiny. Proto SET OF CHAR vždy zabere 32 bytů (možných 256 prvků - jeden bit pro každý prvek. Podobně SET OF Ø .. 1Ø je ekvivalentní s SET OF Ø .. 255 .

hodnotu. Vyjimkou jsou celá čísla, kdy levý operand musí být z intervalu Ø..255.

Výše uvedenou syntaxi aplikujeme při srovnávání řetězců stejné délky, ukazatelů u všech skalárních typů. Množiny můžeme porovnávat použitím >=, <=, <>, nebo =. Ukazatele můžeme porovnávat jenom s použitím = a <>.

2/14 Seznam parametrů



Za dvojtečkou smí být pouze identifikátor typu, jinak je hlášena chyba (* ERROR * 44).

Parametry volané odkazem i hodnotou jsou plně k použití. Procedury a funkce nejsou dovoleny jako parametry.

2/15 Příkazy

Srovnej syntaxi se syntaxním diagramem na str.2Ø.

Přiřazovací příkaz :

Viz kapitolu 2/7 která informuje o nepřípustném složení přiřazovacího příkazu.

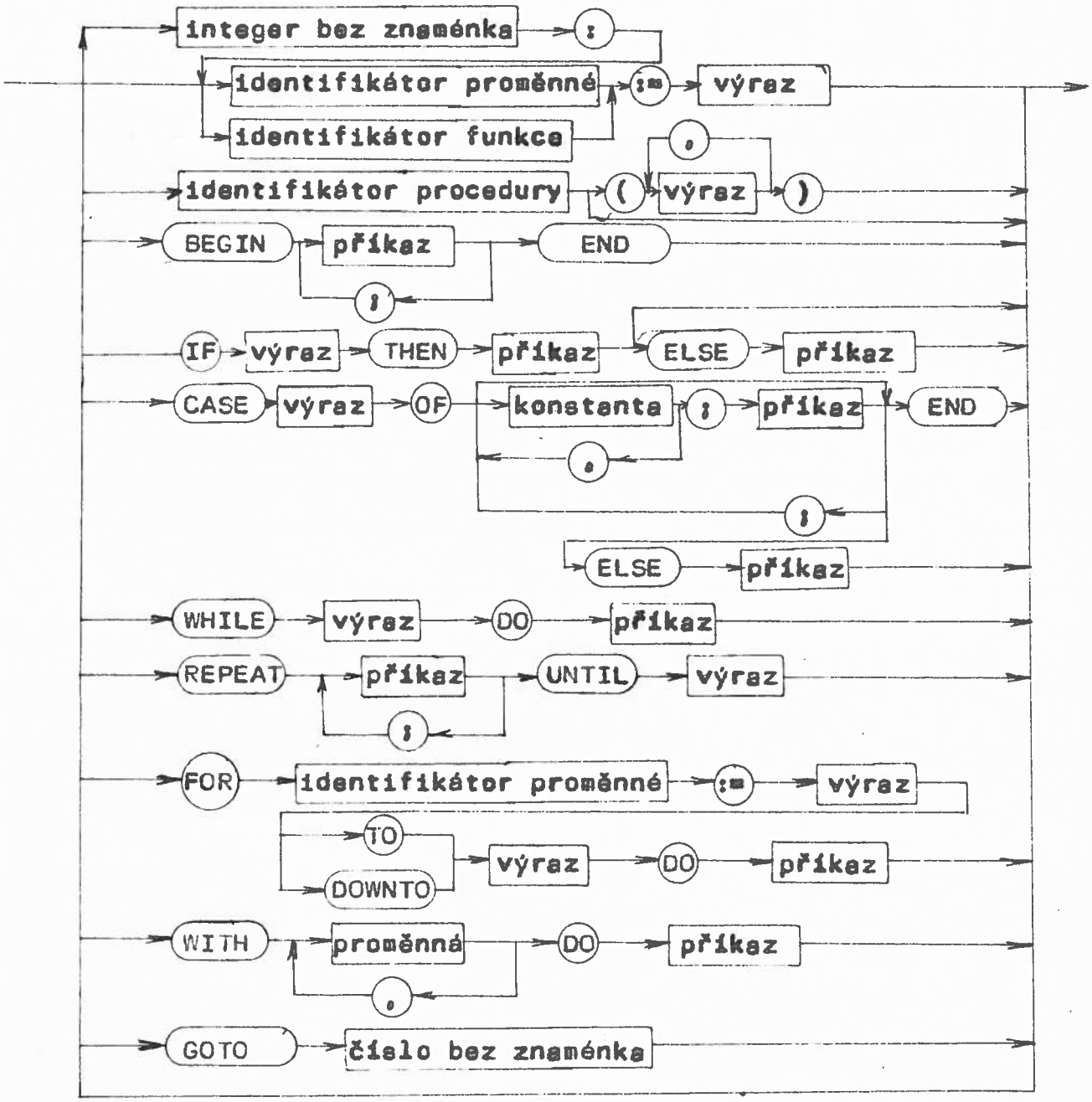
Příkaz CASE :

Není přípustný zcela prázdný zápis CASE, např CASE OF END generuje chybu * ERROR * 13.

Doložka ELSE :

Doložka ELSE, která je alternativou END, je provedena pokud selektor (výraz za CASE) není roven ani jedné konstantě ze seznamu (konstanty před dvojtečkou). Pokud je k ukončení příkazu CASE použito END a selektor není roven žádné konstantě,

PŘÍKAZ



pokračuje program příkazem za END.

Příkaz FOR :

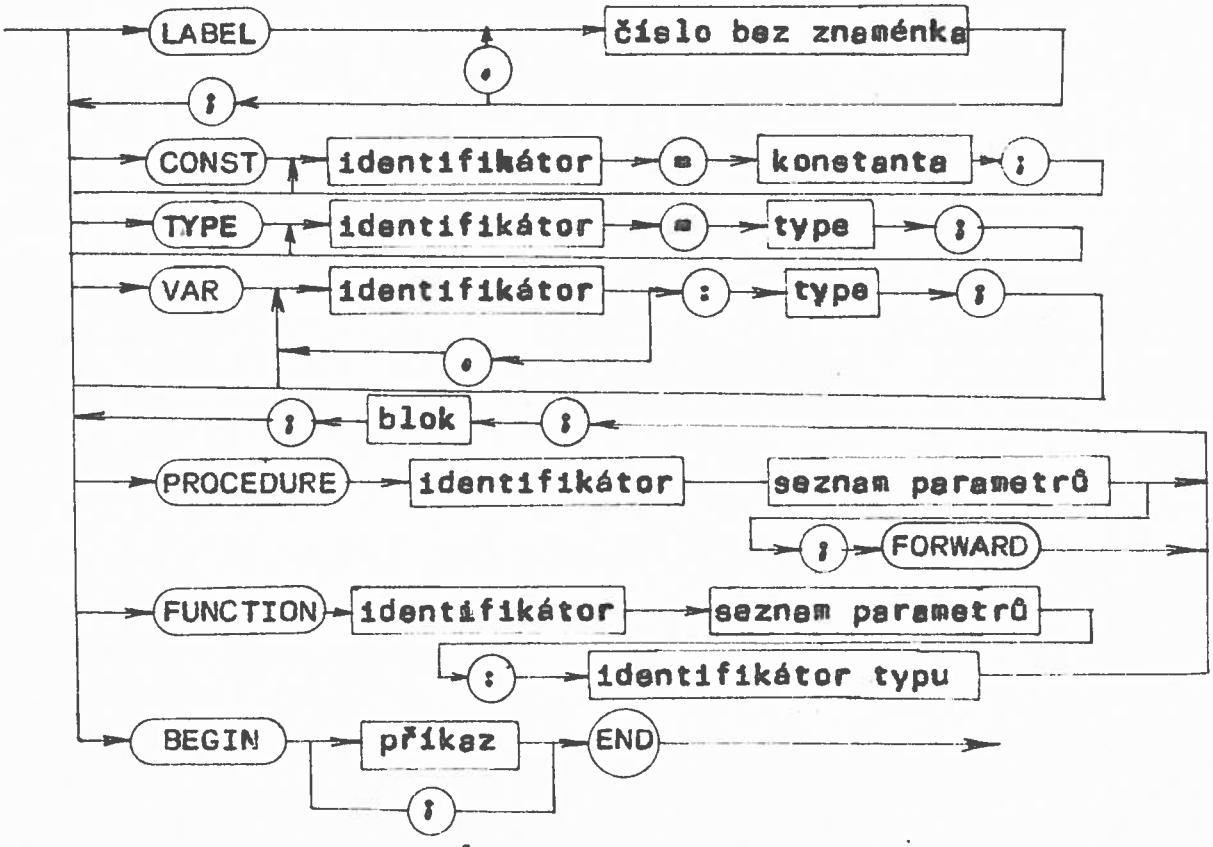
Řídící proměnná příkazu FOR může být nestrukturovanou proměnnou nikoliv parametrem. To je něco mezi Jensen/Wirth a normou ISO.

Příkaz GOTO :

Příkaz GOTO je možný, když návěští je ve stejném bloku a příkaz GOTO je na stejné úrovni.

Návěští musí být deklarováno (použitím rezervovaného slova LABEL) v bloku, kde bude použito a obsahuje jen čtyři číslice. Když je příkaz označen návěštím, musí být před příkazem návěští následované dvojtečkou.

2/16 Programový blok



Poznámka : Když je deklarován soubor proměnných, může na zvláštní přání následovat konstanta v rozsahu 1 až 255 umístěná v hranatých závorkách. Konstanty specifikují velikost registru, který může být použit po 128 znacích na jednotku. Např. jestliže požadujete pole, file[1] a to má být registr 2KB (2048 znaků), ukážeme Vám zde příklad deklarace :

```
VAR   filel : FILE OF CHAR [16] ;
```

nebo

```
CONST filesize = 16;
VAR filel : TEXT [filesize] ;
```

Poznámka : Soubory nejsou v HP 4 implementovány.

Odkazy FORWARD :

V "Pascal User Manual and Report" v kapitole 11.Ø.1 mphou mít procedury a funkce odkaz před deklarací pomocí rezervovaného slova FORWARD. Například :

```
PROCEDURE a(y:t) ; FORWARD;        (procedura a deklarace)
PROCEDURE b(z:t) ;                 (předem tohoto příkazu)
BEGIN
....
a(p);                              (odkaz na proceduru a)
....
END;
PROCEDURE a;                       (nyní deklarace procedury a)
BEGIN
....
b(q);
....
END;
```

Všimněte si, že parametry a typ výsledku je deklarován před slovem FORWARD a není opakován v hlavní deklaraci procedury. Pamatujte si, že FORWARD je rezervované slovo.

2/17 Program

(PROGRAM) [identifikátor] (;) [blok] (END) (.)

Jelikož nejsou implementovány soubory, nemá program žádné formální parametry.

KAPITOLA 3

PŘEDDEFINOVANÉ IDENTIFIKÁTORY

3/1 Konstanty

MAXINT označuje největší celé číslo např. 32767

TRUE, FALSE konstanty typu BOOLEAN

3/2 Typy

INTEGER viz kapitolu 2/3

REAL viz kapitolu 2/3

CHAR Plně rozšířený soubor kódů znaků ASCII, 256 prvků

BOOLEAN (TRUE, FALSE) se používají k zápisu výsledků logických operátorů a porovnávání

3/3 Procedury a funkce

3/3/1 Procedury vstupu a výstupu

3/3/1/1 WRITE

Procedura WRITE se používá pro výstup dat na obrazovku nebo tiskárnu. Když je tisknut jednoduchý výraz typu znak, WRITE (o), prochází osmibitová hodnota, reprezentující hodnotu výrazu o na obrazovku nebo tiskárnu.

Poznámka :

CHR(8)	krok zpět na obrazovce (backspace)
CHR(12)	maže obrazovku nebo dává novou stránku tisku
CHR(13)	provede "návrat vozíku" a posun řádků
CHR(16)	nastaví přímý výstup na tiskárnu je-li používaná obrazovka nebo naopak

Obecné úvahy :

WRITE (P1,P2,.... Pn) je ekvivalentní

BEGIN WRITE (P1); WRITE (P2); ...;WRITE (Pn); END

parametry tisku P1, P2 až Pn mají následný formát :

kde e, m a n jsou výrazy ale H je konstanta (literál)

⟨e⟩ nebo ⟨e:m⟩ nebo ⟨e:m:n⟩ nebo ⟨e:m:H⟩

Zde máte na prostudování pět případů :

1. e je typu integer: je možné buď ⟨e⟩ nebo ⟨e:m⟩ Celočíselný výraz proměnné e je převeden na řetězec znaků a přídavných mezer. Délka řetězce může být zvětšena (přidáním vedoucích mezer) použitím m, které specifikuje celkový počet znaků pro výstup. Jestliže m není dostatečně velké pro e, nebo není m zadáno, provede se výstup a m je ignorováno. Všimněte si, že je-li m zadáno tak, že odpovídá délce znakové reprezentace e bez mezer, potom žádné mezery nejsou na výstupu doplněny.

2. e je typu integer možného formátu ⟨e:m:H⟩

V tomto případě je e na výstupu v hexadecimálním tvaru. Jestliže m=1 nebo m=2 pak tato hodnota proměnné (e MOD 16↑m) má na výstupu přesně m znaků.

Jestliže m=3 nebo m=4 potom je na výstupu plná hodnota e v hexadecimální soustavě rozšířena na čtyři znaky.

Jestliže m>4 potom vedoucí mezery budou vloženy před plnou hexadecimální hodnotu e.

Úvodní nuly budou doplněny podle požadavků. Například :

WRITE (1Ø25:m:H)

pro	m=1	je výstup	1
	m=2		Ø1
	m=3		Ø4Ø1
	m=4		Ø4Ø1
	m=5		_Ø4Ø1

3. e je typu real. Použité formáty jsou :

⟨e⟩ nebo ⟨e:m⟩ nebo ⟨e:m:n⟩

Hodnota e je převedena na řetězec znaků reprezentující reálné číslo. Formát zobrazení je určen pomocí n. Jestliže n není zadáno, potom je výstup ve vědecké notaci s mantisou a exponentem. Záporné číslo má místo mezery před mantisou záporné znaménko. Číslo má vždy nejméně jedno a nejvýše pět desetinných míst, exponent má vždy znaménko + nebo -.

To znamená, že minimální délka řetězce pro zobrazení ve vědecké notaci je osm znaků.

Jestliže m je menší než 8 je vždy zobrazeno plných 12 znaků.

Jestliže m je větší nebo rovno 8 je vždy potom je zobrazeno jedno nebo více desetinných míst až do maxima pěti (m=12), pro m větší než 12 se předřazují mezery před číslo. Například :

WRITE(-1,23E 1Ø:m);

pro	m=7	je výstup	-1,23ØØØE+1Ø
	m=8		-1,2E+1Ø
	m=9		-1,23E+1Ø
	m=1Ø		-1,23ØE+1Ø
	m=11		-1.23ØØE+1Ø
	m=12		-1.23ØØØE+1Ø
	m=13		-1.23ØØØE+1Ø

Při použití formátu ⟨e:m:n⟩ je číslo s pevnou desetinnou tečkou napsáno s n specifikovanými místy. Mezery před číslem, když délka pole m je dostatečně velká.

Jestliže n je rovno nule, je na výstupu celé číslo (integer).

Jestliže e je příliš velké bude výstup ve vědecké notaci a délka pole bude specifikována m (viz výše).

Například :

WRITE(1E2:6:2)	dává	1ØØ.ØØ
WRITE(1E2:8:2)		_1ØØ.ØØ
WRITE(23.455:6:1)		__23.5
WRITE(23.455:4:2)		_2.3455E+Ø1
WRITE(23.455:4:Ø)		__23

4. e je typu znak nebo řetězec.

Buď <e> nebo <e:m> lze použít pro znak nebo řetězec znaků, minimální délka pole pro výstup je 1 (pro znak) a délka řetězce (podle typu řetězce). Předřazené mezery se vkládají při dostatečně velkém m.

5. e je typ boolean.

Můžeme použít buď <e> nebo <e:m> . Výstupem je řetězec 'TRUE' nebo 'FALSE' v závislosti na hodnotě e. Přitom minimální šířka výstupu bude 4 nebo 5 znaků.

3/3/1/2 WRITELN

Výstup WRITELN dává novou řádku. Je to ekvivalent WRITE(CHR(13)). Všimněte si, že zahrnuje posuv řádků.

WRITELN(P1,P2,...,P3); je ekvivalent

BEGIN WRITE(P1,P2, ... ,P3); WRITELN END;

3/3/1/3 PAGE (stránka)

Procedura PAGE je ekvivalent WRITE(CHR(12)), v tomto případě je obrazovka vymazána nebo tiskárna přejde na začátek nové stránky.

3/3/1/4 READ (čtení)

Procedura READ se používá pro vstup dat z klávesnice. Děje se tak přes vyrovnávací paměť, ta je zpočátku prázdná (nehledě na označení začátku a konce řádky). Jakýkoliv přístup do této paměti se děje nahlednutím do okénka, ve kterém můžeme najednou vidět jen jeden znak. Okénko postupuje vyrovnávací pamětí, když je nastaveno na konec řádky, potom dříve než je operace ukončena, bude čtena nová řádka z klávesnice. Čtení řádky z klávesnice je řízeno řídícími klávesami, podrobně je můžete poznat v kapitole 1. Dále :

READ(V1,V2, ... ,Vn); je ekvivaletní

BEGIN READ(V1);READ(V2); ... ;READ(Vn) END;

kde V1,V2 atd budou typu znak, řetězec, integer a real.

Příkaz READ(V) má rozdílný efekt závislý na typu V. Zde jsou čtyři možnosti :

1. V je typu znak.

V tomto případě READ(V) jednoduše čte znak ze vstupní vyrovnávací paměti a přiřadí ho V. Jestliže v textovém okénku vyrovnávací paměti je uložen znak konce řádky (znak (CHR(13)), potom funkce EOLN se vrátí na hodnotu TRUE a je čtena nová řádka textu z klávesnice. Když operace čtení bude znovu volána, textové okénko bude nastaveno na začátek nové řádky.

Důležitá poznámka : Všimněte si, že na začátku programu je EOLN = TRUE. To znamená, že je-li první READ typu znak potom je vrácena hodnota CHR(13) a teprve následující READ čte první znak uživatelem vložené řádky.

Podívejte se také na proceduru READLN uvedenou dále.

2. V je typu řetězec.

Použitím READ můžeme číst řetězec znaků a v tomto případě bude čtena serie znaků až počet znaků definující řetě-

zec znaků bude přečten, nebo EOLN bude TRUE. Jestliže řetězec není během čtení naplněn (to jest jestliže konec řádky bude dřív než bude přiřazen celý řetězec), potom je řetězec doplněn znaky null CHR(Ø) - toto programátorovi umožňuje ohodnotit délku přečteného řetězce.
Poznámka k bodu 1. platí i zde.

3. V je typu integer (celé číslo)

V tomto případě je čtena série znaků reprezentující celé číslo (definované v kapitole 2/3). Všechny předcházející mezery a znaky konec řádky jsou přeskočeny. (To znamená, že je-li první READ typu integer , potom je číslo ihned správně přečteno.)

Jestliže čtené číslo má absolutní hodnotu vyšší než MAXINT (32767) potom je programové chyba "Number too large" a skončí běh programu.

Jestliže první znak po mezeře a znaku začátek-konec řádky má být přeskočen, bez čísla nebo znaménka (+ nebo -), je programová chyba "Number expected" a z touto zprávou se program zastaví.

4. V je typu real (relné číslo)

Zde je číslo čteno jako série znaků reprezentujících reálné číslo eplňující podmínky v kapitole 2/3. Vedoucí mezery a znak konce řádky jsou přeskočeny, jako u celého čísla musí být první následující znak znaménko nebo číslo.

Jestliže je čtené číslo příliš velké nebo malé (viz kapitolu 2/3.) bude zpráva o chybě "Overflow". Jestliže je dáno E bez následujícího znaménka nebo čísla, je generována zpráva o chybě "Exponent expected", desetinná tečka bez následujícího čísla dá chybnou zprávu "Number expected".

Reálná čísla stejně jako celá čísla jsou čtena bezprostředně, viz předcházející body 1. a 3.

3/3/1/5 READLN

READLN(V1,V2, ... ,Vn); je ekvivalentní

BEGIN READ(V1,V2, ... ,Vn); READLN END;

READLN jednoduše čte novou řádku do vyrovnávací paměti a klávesnice. Při čtení do vyrovnávací pam. můžete použít různé řídící klávesy, podrobnosti v kapitole 1. Takže EOLN se stane FALSE je-li provedeno READLN a následující řádka není prázdná.

READLN můžete použít k přeskočení prázdné řádky, která je ve vyrovnávací paměti na začátku provádění programu. To je užitečné, pokud na začátku programu budete číst text, ale ale není to nutné, když čtete čísla.

3/3/2 Vstupní funkce

3/3/2/1 EOLN

Booleovská funkce EOLN se vrací s hodnotou TRUE, když poslední čtený znak je konec řádky (CHR(13)), jinak tato funkce nabývá hodnoty FALSE.

3/3/2/2 INCH

Funkce INCH prohlédne klávesnici, v případě stisklé klávesy se vrátí se znakem reprezentujícím stisklou klávesu. Jestliže není stisknuta žádná klávesa, vrátí se se znakem CHR Ø . Tato funkce má výsledek typu znak.

3/3/3 Funkce převodů

3/3/3/1 TRUNC(X)

Parametr X musí typu real nebo integer a při návratu TRUNC je výsledkem celá část X, menší nebo rovna X je-li X

kladné, případně větší nebo rovna X je-li X záporné. Například :

TRUNC(-1.5) vrací -1 TRUNC(1.9) vrací 1

3/3/3/2 ROUND(X)

X musí být typu real nebo integer. Tato funkce zaokrouhluje X na celé číslo (podle základních pravidel zaokrouhlování). Například :

ROUND(-6.5) = -6 ROUND(11.7) = 12

ROUND(-6.51) = -7 ROUND(23.5) = 24

3/3/3/3 ENTIER(X)

X musí být typu real nebo integer. Výsledkem ENTIER je celá část X menší nebo rovna X pro všechna X. Například :

ENTIER(-6.5) = -7 ENTIER(11.7) = 11

Poznámka : ENTIER není ve standartu Pascalu, je to ekvivalent funkce INT v BASICu. Je využitelná pro napsání rychlého programu pro mnoho matematických aplikací.

3/3/3/4 ORD(X)

X může být skalárního typu s vyjímkou typu real, výsledkem je celé číslo reprezentující pořadové číslo X uvnitř definovaného souboru pro X; jestliže X je typu integer pak ORD(X) = X, toto obvykle obcházíme. Například :

ORD("a") = 97 ORD("K") = 75

3/3/3/5 CHR(X)

X musí být typu integer. Výsledkem CHR je znak mající stejnou hodnotu v ASCII. Například :

CHR(49) = "1" CHR(91) = "["

3/3/4 Aritmetické funkce

Funkce v tomto odstavci vyžadují argument typu real nebo integer.

3/3/4/1 ABS(X)

Výsledkem je absolutní hodnota X. Například ABS(-4.5) dává 4.5. Výsledek je stejného typu jako argument X.

3/3/4/2 SQR(X)

Výsledkem je X * X, to je druhá mocnina X. Výsledek je stejného typu jako X.

3/3/4/3 SQRT(X)

Výsledkem je druhá odmocnina X. Výsledek je typu real. Jestliže je argument X záporný, je signalisována chyba "Mathe Call Error".

3/3/4/4 FRAC(X)

Výsledkem je desetinná část X; FRAC(X) = X - ENTIER(X). Stejně jako ENTIER je tato funkce výhodná pro napsání mnoha rychlých matematických programů. Například :

FRAC(1.5) = 0.5 FRAC(-12.56) = 0.44

3/3/4/5 SIN(X)

Výsledkem je sinus X, kde X je v radianech. Výsledek je vždy typu real.

3/3/4/6 COS(X)

Výsledkem je cosinus X, kde X je v radianech. Výsledek je vždy typu real.

3/3/4/7 TAN(X)

Výsledkem je tangent X, kde X je v radianech. Výsledek je vždy typu real.

3/3/4/8 ARCTAN(X)

Výsledkem je úhel v radiánech pro tangent hodnoty X. výsledek je vždy typu real.

3/3/4/9 EXP(X)

Výsledkem je e^X kde e = 2.71828. Výsledek je typu real.

3/3/4/1Ø LN(X)

Výsledkem je přirozený logaritmus (o základu e) z X. Výsledek je typu real. Jestliže X je menší nebo rovno Ø , je hlášena chyba "Mathe Call Error".

3/3/5 Další předdefinované procedury

3/3/5/1 NEW(p)

Procedura NEW(p) vymezí prostor pro dynamickou proměnnou. Proměnná p musí být typu ukazatel a po provedení NEW(p) bude p obsahovat adresu nově vzniklé dynamické proměnné. Typ této dynamické proměnné je toho typu, který byl uveden při deklaraci ukazatele p.

Přístup k dynamické proměnné získáme napsáním p↑ - - viz příklady v doplňku 4.

K odstranění dynamické proměnné se používají procedury MARK a RELEASE (viz dále).

3/3/5/2 MARK(v1)

Tato procedura zaznamená do ukazatele V1 stav oblasti

pro všechny dynamické proměnné. Stav, který je v okamžiku provedení MARK(v1) obnovíme volání procedury RELEASE v1. Obnovení stavu znamená, že po RELEASE budou existovat jen ty dynamické proměnné, které existovaly již před MARK. Ty, které vznikly až po MARK budou vymazány.

Je důležité, že hodnoty dynamických proměnných, které budou ponechány, MARK ani RELEASE nezmění.

Proměnná v1 je libovolný ukazatel, ale tato proměnná se již nesmí použít v NEW.

Příklady použití MARK a RELEASE jsou v doplňku 4.

3/3/5/3 RELEASE(v1)

Tato procedura uvolní místo v oblasti dynamických proměnných. Je obnoven stav, který byl před provedením MARK v1 , takto se efektivně zruší dynamické proměnné vytvořené po provedení procedury MARK. Při používání je třeba být pozorný.

Viz výše a doplněk 4, kde je více podrobností.

3/3/5/4 INLINE(C1,C2,C3, ...)

Tato procedura dovoluje použití strojového kódu Z80, vloženého do programu v Pascalu; hodnoty (C1 MOD 256, C2 MOD 256, C3 MOD 256, ...) jsou vleženy do přeloženého programu, aktivní hodnotu čítače adres nastaví kompilátor.
C1,C2 a C3 atd. jsou celočíselné konstanty v libovolném počtu.

Příklady použití INLINE jsou v doplňku 4..

3/3/5/5 USER(V)

USER je funkce s celočíselným argumentem V . Tato procedura volá přímo paměť a adresu V. Protože HISOFT PASCAL 4 ukládá celé číslo ve dvojkovém doplňkovém formátu (viz dopl-

něk 3.) proto, aby jste se mohli odkazovat na adresy větší než # 7FFF (32767 decimálně), musíte použít záporné hodnoty V. Například # CØØØ je -16384 a USER(-16384) vyvolá skok na adresu #CØØØ.

Avšak, když je V konetanta, je vhodné použít hexadecimální soustavy.

Přímé volání programu Z8Ø ukončíme instruvkcí RET(C9), přitom musíme zachovat obsah registru IX.

3/3/5/6 HALT

Tato procedura zastaví běh programu a podá zprávu "Halt at PC=XXXX" kde XXXX je hexadecimální adresa paměti ukazující kde HALT způsobí zastavení.

Společně s výpisem kompilátoru může být HALT použito k zjištění, kterou ze dvou či více větví program prochází.

Toto můžete běžně používat při ladění programů.

3/3/5/7 POKE(X,V)

POKE uloží výraz V do paměti počítače počínaje adresou X. Kde X je typu integer. Viz kapitola 3/3/5/5 výše, kde zjistíte jaké celé číslo reprezentuje adresu paměti.

Například :

POKE(#6ØØØ,"A")	umístí #41 na adresu #6ØØØ
POKE(-16384,3.6E3)	umístí ØØ ØB 8Ø 7Ø (hex) od adresy #CØØØ

3/3/5/8 TOUT(NAME, START, SIZE)

Procedura TOUT se používá pro uložení proměnných na pásku. První parametr je typu ARRAY[1..8] OF CHAR a je to jméno nahraného pole bytů o délce SIZE z paměti od počáteční adresy START. Oba dva další parametry jsou typu integer.

Například : ukládáme-li proměnnou V pod jménem VAR, použijeme :

TOUT('VAR ',ADDR(V),SIZE(V))

Použití adresy dává větší pružnost použití než využívání scho-pnosti nahrávat pole. Například, jestliže systém má mapu obrazovky, můžeme celou obrazovku nahrát přímo.

Viz doplněk 4. s příklady použití TOUT.

3/3/5/9 TIN(NAME,START)

Tato procedura se používá ke čtení proměnných a pod. z pásky, byly-li uloženy pomocí TOUT. NAME je typu ARRAY [1..8] OF CHAR a START je typu integer. Na pásce se vyhledá pole se jménem NAME, které uloží do paměti od adresy START. Počet bytů je udán na pásce (byl tam nahrán TOUT). Například proměnnou uloženou v odstavci 3/3/5/8. přečteme takto :

TIN('VAR ',ADDR(V))

Poněvadž zdrojové texty jsou napsány editorem používající stejný formát jako TIN a TOUT, může být TIN použito k zpracování zdrojového textu znakových polí ARRAY OF CHAR. (Viz HP4T Alternation Guide) Viz příklady v doplňku 4.

3/3/5/10 OUT(P,C)

Tato procedura se používá pro přímý přístup k výstupním portům Z8Ø bez použití INLINE. Celočíselná hodnota parametru "P" je vložena do registrového páru BC a parametr "C" typu znak je uložen do A registru, pak se provede instrukce assembleru OUT(C),A . Například :

OUT(1,'A') pošle znak 'A' na první port Z8Ø

3/3/6 Další předdefinované funkce

3/3/6/1 RANDOM

Výsledkem je celé pseudonáhodné číslo v rozsahu Ø až 255. Třebaže je tento program velmi rychlý dává slabý výsledek, když použijeme opakování smyčky bez vložení I/O operací.

Požadujete-li lepší výsledky, napište si program (v Pascalu nebo strojovém kódu) přizpůsobený pro jednotlivé použití.

3/3/6/2 SUCC(X)

X je skalárního typu s vyloučením reálných čísel. Výsledkem je následující X. Například :

SUCC('A') = 'B' SUCC('6') = '7'

3/3/6/3 PRED(X)

X musí být skalárního typu s vyloučením reálných čísel. Výsledkem funkce je předcházející X. Například :

PRED('j') = 'i' PRED('TRUE') = FALSE

3/3/6/4 ODD(X)

X musí být typu integer. Výsledek je TRUE pro liché X a FALSE pro sudé X.

3/3/6/5 ADDR(V)

Tato funkce má parametr identifikátor proměnné libovolného typu, výsledkem je celé číslo, které je adresou proměnné s identifikátorem V , v paměti. Informace o umístění proměnných a programu v HP4 jsou doplňku 3. Příklady použití jsou v doplňku 4.

3/3/6/6 PEEK(X,T)

První parametr této funkce je typu integer a udává adresu paměti (viz odstavec 3/3/5/5.). Druhým argumentem je typ a tento typ je výsledkem funkce.

PEEK se používá k vyhledání dat v paměti počítače a výsledkem je libovolný typ.

PEEK a POKE (opak PEEK) operují nad daty, mění vlastní vnitřní reprezentaci v HISOFT PASCALU. Podrobnosti jsou v doplňku 3. Například jestliže v paměti od adresy #5000 jsou uloženy hodnoty : 50, 61, 73, 63, 61, 6C (hexadec.) pak :

WRITE(PEEK(#5000,ARRAY [1..6] OF CHAR)) dává 'Pascal'

WRITE(PEEK(#5000,CHAR)) dává 'P'

WRITE(PEEK(#5000,INTEGER)) dává 24912

WRITE(PEEK(#5000,REAL)) dává 2,460276E+29

Více podrobností o reprezentaci zápisů v HISOFT PASCALu je v doplňku 3.

3/3/6/7 SIZE(V)

Parametrem této funkce je proměnná. Celo číselný výsledek udává délku obsazené paměti v bytech.

3/3/6/8 INP(P)

Používá porty Z80 přímo bez použití procedury INLINE. Hodnota celočíselného parametru P (adresa portu Z80) se vloží do registrového páru BC a výsledek je znak po provedení assemblerové instrukce IN A,(C).

KAPITOLA 4
POZNÁMKY A VARIANTY KOMPILÁTORU

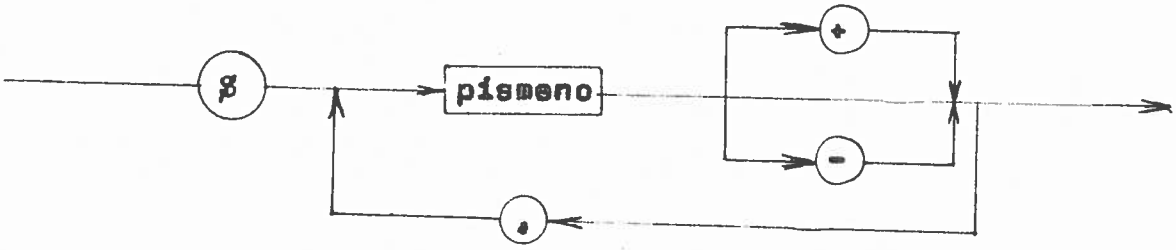
4/1 Poznámky

Poznámky mohou být vždy mezi dvojicí rezervovaných slov, čísel, identifikátorů a speciálních symbolů. Viz doplněk 2.

Poznámka začíná znakem { nebo dvojicí znaků(*, není-li bezprostředně následující znak $, jsou všechny znaky ignorovány až do dalšího znaku } nebo dvojice znaků *). Jestliže kompilátor nalezne znak $ ví, že je to série variant kompilátoru (viz. následující), a potom je provádí až do nalezení } nebo *), které přeskočí.

4/2 Varianty kompilátoru

Syntax specifikující varianty kompilátoru :



Následují možné varianty :

Varianta L

Řídí výpis programového textu a cílového kódu adresovaných kompilátorem.

Je-li L+ potom je nastaven plný výpis.

Je-li L- je vypsána řádka pouze, když je v ní zjištěna chyba. Standartní hodnota je L+.

Varianta O

Kontroluje zda nedošlo k přetečení během celočíselného sčítání a odčítání. Při aritmetických operacích násobení a dělení kontroluje přetečení vždy.

Je-li O+ potom kontroluje celočíselné sčítání a odčítání.
Je-li O- potom výše uvedená kontrola není prováděna.
Standartní hodnota je O+.

Varianta C

Kontroluje nebo nekontroluje klávesnici v průběhu provádění cílového programu.
Je-li C+ potom stisknutí EDIT zastaví program pomocí HALT, zpráva , viz odstavec 3/3/5/6.

Touto kontrolou začínají všechny smyčky funkcí a procedur. Této příležitosti můžeme využít pro detekci smyčky atd. není vhodné se vzdát této možnost v průběhu ladění programu. Zablokování provedeme jestliže si přejeme rychle běžící cílový program.
Je-li C- potom není výše uvedená kontrola prováděna.
Standartní hodnota C+.

Varianta S

Určuje, zda má testovat přetečení zásobníku.
Je-li S+ potom začátek každé procedury a funkce vyvolá kontrolu zda je pravděpodobné přeplnění zásobníku v tomto bloku. Jestliže je programový zásobník přeplněn dynamickou proměnnou nebo programem, následuje zpráva "Cut of RAM at PC=XXXX" a běh programu se zastaví. Přirozeně, že to není snadno ovladatelné, jestliže procedura má používat velký vnitřní zásobník, pak program sám od sebe může havarovat.
Naproti tomu jestliže funkce obsahuje velmi malý zásobník průběžně se zvětšující, pak je možné, že zastavení touto funkcí bude zbytečné.
Je-li S- potom kontrola zásobníku není prováděna,
Standartní hodnota S+.

Varianta A

Řídící kontrolu zabezpečující index pole v mezích hranic specifikovaných v deklaraci pole.
Je-li A+ a index pole je příliš velký nebo naopak příliš malý, pak následuje zpráva "Index too high" nebo "Index too low" a běh programu se zastaví.
Je-li A- potom se kontrola neprovádí.
Standartní hodnota A+.

Varianta I

Při používání 16-ti bitového dvojkového doplňku celočíselné aritmetiky může nastat přetečení při provádění operací >, < , >= nebo <= , liší-li se argumenty o více než MAXINT (32767). Jestliže se tak stane, výsledek porovnávání bude chybný. Toto obvykle nepůsobí potíže avšak, když musíme použít srovnání takových čísel, potom použití I+ zabezpečí, že výsledek srovnávání bude správný. Stejná situace může nastat při aritmetice s reálnými čísly v tomto případě nastane přetečení jestliže je rozdíl argumentů asi 3.4E38, tomu se nelze vyhnout.
Je-li I- potom není výše uvedená kontrola prováděna.
Standartní hodnota je I+.

Varianta P

Když použijeme variantu P , změní se zařízení na kterém probíhá výpis protokolu o překladu.
Například : jestliže je na obrazovce, začne být tisknut na tiskárně nebo naopak.
Standartní hodnota : používá obrazovku.
Poznámka ; tato verze není doplněna + nebo - .

Varianta F

Po tomto písmenu musí následovat mezera a jméno pole s osmi znaky. Má-li jméno pole méně než osm znaků je nutné jej doplnit mezerami.

V případě volby této varianty se zahrne zdrojový text v Pascalu uložený na pásce, toto lze využít,, na přání programátora, k sestavení knihovny pro nejvíce používané procedury a funkce v programech na pásce. Program uložíme pomocí povelu "W" vestavěného v editoru.

V nejvyšším systémovém výpisu použijeme variantu L- jinak kompilátor nebude překládát dost rychle. Například :

{$L-,F MATRIX začlenění text souboru MATRIX z pásky}

Při psaní velmi rozsáhlých programů není dost místa v paměti pro zdrojový text a cílový kód současně. Možnost překladu takových programů vám nechrání použití varianty "F" a pásky, potom je současně ukládáno v RAM pouze 128 bytů a zbude nám dostatek místa pro cílový kód.

Varianty kompilátoru mohou být použity samostatně. Takto bude rychlejší a kompaktnější odlaďovací sekce kódů pro nejdůležitější nutnou kontrolu, zatímco ostatní řídící kódy se zadrží.

K A P I T O L A 5
I N T E G R Á L N Í E D I T O R

5/1 Úvod do editoru

Všechny varianty HISOFT PASCALu doplňuje editor, jde o jednoduchý, řádkově orientovaný editor určený pro Z8Ø.

Text je v paměti uložen v kompaktní formě, počet úvodních mezer v řádce je uložen jako jeden znak; počátek řádky a všechny rezervované slova HP4T jsou zaznamenána znaky. Toto vede k typickému zkrácení textu asi o 25%.

Poznámka : V této kapitole dává klávesa DELETE řídící kód CH (vymaž poslední znak), což je přirozenější.

Když je editor vložen z pásky automaticky se zobrazí zpráva :

Copyright Hisoft 1982

All right reserved

bezprostředně následovaná editorovým kurzorem " > "
A hned můžeme vkládat povelovou řádku v tomto formátu :

C N1, N2 , S1, S2

následujícím ENTER, kde :

C je povel, který má být proveden (viz kapitola 5/2)

N1 je číslo v rozsahu 1 až 32767 včetně

N2 je číslo v rozsahu 1 až 32767 včetně

S1 je řetězec znaků s maximální délkou 2Ø znaků

S2 je řetězec znaků s maximální délkou 2Ø znaků

Čárka je použita jako oddělovač různých argumentů (ačkoliv toto může být změněno - viz povel "S") a mezery jsou ignorovány, kromě mezer uvnitř řetězce. Žádný z argumentů není příkaz ačkoliv některé z povelů (například povel "D"elete) mohou být specifkovány bez N1 a N2. Editor si pamatuje předcházející řetězec a číslo, které jste vložili a použije shod-

ný formát hodnot, jestliže není specifikován další argument uvnitř povelové řádky. Hodnoty N1 a N2 mají zpočátku hodnotu 1Ø a řetězec je prázdný. Jestliže vložíme nepřípustný povel do řádky třeba např.: F-1,1ØØ,HELLO potom je řádka ignorována a zobrazí se zpráva "Pardon?" a řádku musíme opravit např.: F1,1ØØ,HELLO. Stejná chyba bude jestliže délka S2 bude větší než 2Ø znaků. Jeszliže S1 je delší než 2Ø znaků, potom všechny znaky navíc budou ignorovány.

Povely mohou být vkládány malými i velkými písmeny.

Při vkládání povelové řádky, můžete použít všechny důležité řídící kláveey popsané v kapitole 1., například ← vymaže celou řádku.

Následující odstavce doplní použití různých povelů v editoru. Všiměte si, že kdykoliv je argument uzavřen mezi symboly "< >" potom musí být tento argument zadán v povelu.

5/2 Povely editoru

5/2/1 Vkládání textu

Text bude uložen do textového souboru, buď , vložením čísla řádky, mezery a textu (jako v BASICu) nebo povelem "I". Všimněte si, že napíšete-li číslo řádky a následuje ENTER (není žádný text), tato řádka bude vymazána z textu (jestliže existuje). Kdykoliv vkládáme text můžeme použít řídící funkce ← (vymazání celé řádky), → (jdi na příští TAB pozici), EDIT(návrat do povelové smyčky). Klávesa DELETE způsobí zrušení předcházejícího znaku (ale ne na začátku textové řádky). Text se vkládá do vyrovnávací paměti HP4T a jestliže se tato paěmť naplní, potom před vložením dalšího textu musíte preventivně použít DELETE nebo ← pro uvolnění místa ve vyrovnávací paměti.

Povel I n,m (Insert)

Tento povel vyvolá mod automatického vkládání. Začínáte vkládat od čísla n , zvyšujícím se o krok m . Po zobrazení čísla řádky můžete vkládat text, použít různých řídících kódů a přejete-li si ukončíte řádku ENTER. Tento mod zrušíte použitím řídicí funkce EDIT (viz kapitola 1. s důležitou doplňující poznámkou).

Vložíte-li řádku bez ohledu na to, že již existuje toto číslo řádky s textem, bude po stisknutí ENTER text na této řádce vymazán a nahrazen textem nové řádky. Dojde-li při automatickém zvyšování čísla řádky k číslu řádky většímu než 32767 potom Insert mod skončí.

Když při psaní textu dojdete na konec řádky na obrazovce bez toho, že překročíte 128 znaků (délka vyrovnávací paměti) tak se obrazovka posune o řádku vzhůru a Vy můžete pokračovat v psaní na následující řádce.

5/2/2 Vypisování textu

Text může být prohlížen pomocí povelu L, počet řádek současně zobrazených je dán (viz doplňující poznámka), ale může být změněn povelem "K".

Povel "L n,m"

Vypisuje aktivní text na displej, počínaje číslem řádky n , do čísla řádky m včetně. Standartní hodnoty pro n a m 1 a 32767. To znamená, že standartní hodnota není brána z předchozího povelu. Pro výpis celého textového pole použijeme "L" bez argumentů. Počet řádek na obrazovce může být ovládán pomocí povelu "K", - potom po vypsání jistého počtu řádek se udělá mezera (jestliže to ještě není řádka m), stiskem řídicí klávesy EDIT se vrátí řízení editoru nebo li-

bovolně stisknutá další klávesa pokračuje ve výpisu.

Povel K n

Nastaví počet řádek výpisu současně zobrazovaných na obrazovce, předtím udělá na obrazovce mezeru pro oddělení předchozího výpisu v "L" modu. Vypočte a uloží hodnotu (n MOD 256). Například : K 5 vytvoří výpis pěti řádek na obrazovce současně.

5/2/3 Úpravy textu

Jednou daný text na řádce je někdy třeba nezbytně trochu pozměnit. Různé povely dokážou řádku vymazat, posunout a přečíslovat.

Povel D (n,m)

Vymaže všechny řádky od n do m včtně textového pole. Když m je menší než n nebo nejsou zadány argumenty, nic se nestane; je to ochrana před chybou a nepozorností. Jedna řádka může být vymazána jestliže m=n; jednodušší je to ovšem mašením čísla řádky následovaným ENTER.

Povel M n,m

Text řádky n bude vložen na řádku m a vymaže text, který tam byl před tím. Všimněte si, že řádka n se nezmění. Tento povel vám dovoluje posunout (přesunout) text i na řádku neexistující.

Povel N (n,m)

Při použití povelu "N" je textové pole přečíslováno. První řádka číslo n bude následována dalšími s krokem m.

Musíme zadat m i n ,v případě kdyby některá řádka měla překročit 32767, zůstane zachováno původní číslování.

Povel F n,m,f,s

Existující textové řádky v rozsahu $n<x<m$ se prohledávají zda obsahují hledaný řetězec f -(find). Nastane-li takový případ zobrazí se celá textová řádka v edit modu - viz následující. Potom můžete použít povelů edit modu k pátrání po následujících případech řetězce "f" v definovaném rozsahu řádek nebo nalezený řetězec nahradit řetězcem s - (substitute) a pátrat po dalším výskytu řetězce f;více podrobnosti dále.
Povel F n,m,f,s může být použit také pro nastavení parametrů n,m,f,s pro jiné povely editoru viz níže.

Povel E n

Edituje řádku číslo n , jestliže neexistuje pak nic nenastane, jinak se řádka překopíruje do vyrovnávací paměti a zobrazí se na obrazovce; řádka číslo n je znova zobrazena pod upravenými řádkami a je v edit modu. Všechny následující úpravy probíhají ve vyrovnávací paměti a ne v textu; proto originál řádky můžeme kdykoliv vyvolat.

V tomto modu se po řádce pohybuje imaginární kurzor (od prvého znaku), a různé podpovely Vám pomáhají a umožňují opravy na řádce. Jsou to následující podpovely :

SPACE - Posuv kurzoru vpravo, zvyšuje ukazatel textu o jednu t.j. místo posledního znaku řetězce, nelze udělat krok na konec řádky.

DELETE - Posun do leva, snižuje ukazatel textu o jednu a

posouvá dopředu následující znaky na řádce. Nemůžete udělat zpětný krok před první znak na řádce.

→ - (řídící funkce) Nastaví textový ukazatel na následující TAB pozici, ale ne na konec řádky.

ENTER - Ukončí editaci, tato řádka se stane platnou

Q - opustí editaci této řádky, t.j. zanech editace ignoruj všechny provedené změny a zanechaná řádka zůsta ne jako byla před začátkem editace.

R - obnov text ve vyrovnávací paměti, t.j. zapomenou se všechny provedené změny na této řádce, bude obnovena původní řádka a editace pokračuje.

L - vypiš zbývající část editované řádky, t.j. zbytek řádky za aktivní pozicí kurzoru. Edit mod zůstává, přes přemístění kurzpru na začátek řádky.

K - zruš znak na aktivní pozici kurzoru

Z - vymaž všechny následující znaky od aktivní pozice kurzoru (včetně) až do konce řádky.

F - Hledá až do nalezení "hledaný řetězec" f předem definovaný uvnitř povelové řádky (viz povel "F" výše). Tento podpovel automaticky ukončí editaci aktivní řádky (zachová změny), jestliže není nalezen jiný "hledaný řetězec" na aktivní řádce (jestliže je nalezen v násl. řádce) (uvnitř dříve specifikovaného rozsahu řádek), potom vetoupí do edit modu řádka, jenž

řetězec obsahuje. Všimněte si, že po úspěšném nalezení je textový kurzor vždy na počátku řádky.

S - nahradí předem definovaným "náhradním" řetězcem s, nalezený "hledaný" řetězec f , nalezený podpovelem F, t.j. v případě nalezení "hledaného"řetězce, toto spolu s povelem F se používá jako krok pro optimalizaci textového souboru v případě nahražování "hledaného" řetězce, řetězcem "náhradním" - viz příklady v kapitole 5/3.

I - vlož znak na aktivní pozici ukazatele. V tomto submodu budete dokud nestisknete ENTER - ten vše vrátí do hlavního edit modu s ukazatelem na pozici posledního vloženého znaku. Uvnitř tohoto submodu lze použít (řídící funkci) DELETE pro případ, kdy znak vlevo od kurzoru má být vymazán z vyrovnávací paměti. Použitím → lze kurzor nastavit na další TAB posici vložením mezer.

X - posune kurzor na konec řádky a automaticky spustí I submod vkládání, uvedený výše.

C - submod změny, dovolí přepsat znaky na aktivní pozici kurzoru a ukazatel posune o jednu vpřed, vy zůstáváte v submodu změny až do stisknutí ENTER, kdy přejdete do edit modu a kurzor je opět na pozici posledního měněného znaku. DELETE uvnitř submodu jednoduše sníží ukazatel ojednu, t.j. posune jej vlevo. Řídící funkce → je bez efektu

5/2/4 Povely pro magnetofon

Text může být ukládán na pásku a přehráván z pásky, použitím povelů "P" a "G".

Povel P n,m,s

Řádky v rozsahu definovaném n a m jsou uloženy na pásku ve formátu HP4T pod jménem specifikovaným řetězcem s. Vzpomeňte si jaký smí být argument v předchozích povelech.

Povel G ,,s

Na pásce je hledáno pole ve formátu HP4T specifikované jménem pole (řetězcem s). Jestliže je nalezeno platné pole HP4T na pásce, ale má jiné jméno je zpráva "Found" následována jménem nalezeného pole na pásce a hledání pokračuje dál. Jakmile je nalezeno správné jméno potom "Found" ukáže označení a pole je přehráno do paměti. Při zjištění chyby během čtení se zobrazí "Checksum error" a čtení se přeruší. Když se to stane, musíte vrátit pásku, opět napsat povel "G" a spustit přehrávání.

Jestliže je řetězec prázdný, potom je do paměti přečteno první pole formátu HP4T, bez ohledu na jeho název.

Prohledávání a čtení z pásky můžete přerušit stisknutím EDIT; což vás vrátí do hlavní smyčky editoru.

Poznámka : Jestliže už je dáno nějaké textové pole, pak textové pole z pásky bude přečteno a připojeno k existujícímu souboru, celý soubor bude přečíslován od řádky 1 s krokem 1.

5/2/5 Překlad a spuštění z editoru

Povel C n

Spouští překlad od řádky číslo n. Není-li číslo řádky specifikováno, je text překládán od první existující řádky. Další podrobnosti jsou v kapitole 1/2.

Povel Y

Spustí přeložený cílový kód, ale jenom v případě, když mezi tím nebyl zdrojový text rozšířen - viz kapit. 1/2.

Povel T n

"T"ranslate (přeložení). Aktivní zdrojový text je přeložen od řádky n (nebo od začátku, není-li specifikováno) a je-li překlad úspěšný, následuje zpráva "Ok?", když odpovíte "Y" bude kompilátorem přeložený cílový program (kód) přesunut na konec, současně se smaže kompilátor a cílový kód bude nahrán na pásku se jménem pole jako dříve definovaný "hledaný" řetězec. Později můžete toto pole přečíst zpět do paměti a automaticky bude spuštěn cílový program. Všimněte si, že cílový kód je přesunut a umístěn na konci, a po povelu T je třeba znova načíst kompilátor i když to není žádný problém, je vhodné přesouvat plně práceschopné programy.

Rozhodnete-li se nepokračovat, nahráváním na pásku, tak jednoduše stiskněte libovolnou jinou klávesu než "Y", okamžitě následuje zpráva "Ok?" a řízení se vrátí editoru se všemi jeho funkcemi, protože cílový kód není přesunut.

5/2/6 Ostatní povely

Povel B

Vrátí řízení operačnímu systému BASICu. Podrobnosti jak se vrátíte do kompilátoru jsou v"HP4T Alteration Guide:

Povel O n,m

Vzpomeňte si jak je text uložen v paměti ve formě znaků, vedoucí mezery jsou zkráceny do znaku jejich počtu a všechna rezervovaná slova HP4T jsou zredukovány na znak.

Nicméně jestliže máte nějak získat text z paměti, snad z jiného editoru, který není ve slovíčkách, použijete povel O , který pro Vás složí slovíčka. Text je čten do vyrovnávací paměti v rozšířené formě a tedy se vrátí do původní formy pole ve formě slovíček, toto proběhne během krátké chvíle. Rozsah řádek musí být specifikován, nebo vložíte alespoň předpokládané hodnoty.

Povel S..d

Dovoluje měnit separátory argumentů v povelové řádce. V editoru je separátorem čárka ","; to lze změnit povelem S ve kterém první znak specifikuje řetězec d. Pamatujte si, že když definujete nový separátor musíte jej používat (i uvnitř případného povelu S) až do další definice. Poznámka : tímto separátorem nemůže být mezera.

Povel X

Zobrazí v šestnáctkové soustavě poslední adresu, na které je ještě uložen překladač. Tuto informaci můžete použít při kopírování překladače.

Povel W n,m,s

Pracuje stejně jako povel P, ovšem nahrává text programu na pásku ve zvláštním formátu, který vyžaduje variatu kompilátoru {$F name} .

Upozornění : Text nahraný povelem W lze přečíst pomocí {$F name} nikoliv však povelem G.

Povel V

Vypíše právě nastavené hodnoty parametrů n,m,f,s pro povely editoru.

5/3 Příklad použití editoru

Napište následující program (s použitím I 1Ø,1Ø) :

```
1Ø PROGRAM BUBLESORT
2Ø CONST
3Ø   Size = 2ØØØ;
4Ø VAR
5Ø   Numbers : ARRAY [1..Size] OF INTEGER;
6Ø   I, Temp : INTEGER;
7Ø BEGIN
8Ø   FOR I:=1 TO Size DO Number [I] := RANDOM;
9Ø   REPEAT
1ØØ     FOR I:=1 TO Size DO
11Ø     Neswape := TRUE;
12Ø     IF Number [I]> Number [I+1] THEN
13Ø       BEGIN
14Ø         Temp := Number [I] ;
15Ø         Number [I] := Number [I+1] ;
16Ø         Number [I+1]:= Temp ;
17Ø         Neswape := FALSE
```

```
18Ø          END
19Ø     UNTIL Neswapes
2ØØ END.
```

V tomto programu jsou tyto chyby

1Ø chybí středník

3Ø zde není skutečná chyba , ale my potřebujeme Size = 1ØØ

1ØØ velikost může být Size-1

11Ø toto má být na řádce 95

19Ø místo Neswapes má být Neswape.

Je deklarována proměnná Numbers , ale všechny odkazy jsou na Number. Nakonec proměnná BOOLEAN Neswape nemá deklaraci.

Tyto chyby odstraníme následujícími postupy :

F6Ø,2ØØ,Number,Numbers	opakovaně použijem povelu S
E1Ø	následuje X ; ENTER ENTER
E3Ø pak	pak ---------- K C 1 ENTER ENTER
F1ØØ,1ØØ,Size,Size-1	následuje povel B
M11Ø,95	"
11Ø	pak ENTER
E19Ø	pak X DELETE ENTER ENTER
65 Neswape : BOOLEAN ;	.
N1Ø,1Ø	přečíslujeme s krokem 10

Doporučujeme Vám přesně provést předcházející příklad, na poprvé se Vám může zdát použití editoru těžkopádné, ale po delším používání si jistě zvyknete.

C H Y B Y

D O P L N Ě K 1

A/1/1 Chyby generované kompilátorem

kód	příčina
1	příliš velké číslo
2	očekáván středník
3	nedeklarovaný identifikátor
4	očekáván identifikátor
5	použijte "=" a ne ":=" při deklaraci konetant
6	očekáváno "="
7	tímto identifikátorem nemůže začínat příkaz
8	očekáváno ":="
9	očekávána ")"
10	chybný typ
11	očekávána "."
12	očekáván činitel (Factor)
13	očekávána konstanta
14	tento identifikátor není konstanta
15	očekáváno "THEN"
16	očekáváno "DO"
17	očekáváno "TO" nebo "DOWNTO"
18	očekávána "("
19	nemůžete napsat tento typ výrazu
20	očekáváno "OF"
21	očekávána ","
22	očekávána ":"
23	očekáván "PROGRAM"
24	očekávaná proměnná, protože parametr je parametr proměnná
25	očekáváno "BEGIN"

26 chybí proměnná ve volání READ

27 nemůžete srovnávat výrazy tohoto typu

28 typ může být buď INTEGER nebo REAL

29 nemůžete číst tento typ proměnné

3Ø tento identifikátor není typ

31 očekával se exponent reálného čísla

32 očekáván skalární výraz (ne numerický)

33 nulové řetězce nejsou dovoleny (užití CHR(Ø))

34 očekávána " ["

35 očekávána "] "

36 index pole musí být skalárního typu

37 očekáváno " .. "

38 při deklaraci pole se očekává "] " nebo ","

39 dolní mez je větší než horní mez

4Ø příliš velká množina (více než 256 možných prvků)

41 výsledek funkce musí být zadán identifikátorem typu

42 v množině se očekává "," nebo "] "

43 v množině se očekává ".." nebo "," nebo "] "

44 typ parametru musí být zadán identifikátorem typu

45 prázdná množina nemůže být prvním činitelem v příkazu, který není přiřazení

46 očekáván skalár (včetně real)

47 očekáván skalár (kromě typu real)

48 neslučitelné množiny

49 nelze použít "<" a ">" k porovnávání množin

5Ø očekáváno "FORWARD","LABEL","CONST","VAR","TYPE" nebo "BEGIN"

51 očekáváno hexadecimální číslo

52 nelze dělat POKE množiny

53 příliš velké pole (větší než 64KB)

54 očekáváno "END" nebo ";" v definici záznamu

55 očekáván identifikátor položky

56 očekáváne proměnná po "WITH"

57 proměnná WITH musí být typu záznam

58 identifikátor položky není asociován s příkazem "WITH"

59 po "LABEL" se očekává integer bez znaménka

60 po "GO TO" se očekává integer bez znaménka

61 toto návěští má špatnou úroveň

62 nedeklarované návěští

63 parametrem SIZE může být jen proměnná

64 můžete použít jen testy rovnosti ukazatelů

67 špatně zadaný formát v proceduře WRITE

68 řetězec nemůže obsahovat znak konce řádky

69 parametr NEW, MARK nebo RELASE musí být proměnnou typu ukazatel

70 parametr ADDR může být jen proměnná

A/1/2 Chybové zprávy při běhu programu

Když je při běhu programu zjištěna chyba, zobrazí se zpráva následovaná "PC=XXXX" , kde XXXX je hexadecimální adresa paměti, kde došlo k zastavení. Často jsou příčiny chyb zřejmé, podívejte se na výpis kompilátoru, podle XXXX uvidíte kde nastala chyba.

Že nedostáváte správné výsledky působí tyto příčiny.

1 Halt (zastavení).

2 Overflow (přetečení)

3 Out of RAM (mimo rozsah RAM)

4 / by zero nebo také DIV (dělení nulou)

5 Index too low (nízký index)

6 Index too high (vysoký index)

7 Mathe Call Error (matematikou vyvolaná chyba)

8 Number too large (příliš velké číslo)

9 Number expected (očekáváno číslo)

1Ø Line too long (příliš dlouhá řádka)

11 Exponent expected (očekáván exponent)

Běh programu je chybou zastaven.

D O P L N Ě K 2

R E Z E R V O V A N Á S L O V A

P Ř E D D E F I N O V A N É I D E N T I F I K Á T O R Y

A/2/1 **Rezervované slova**

AND	ARRAY	BEGIN	CASE	CONST
DIV	DO	DOWNTO	ELSE	END
FORWARD	FUNCTION	GOTO	IF	IN
LABEL	MOD	NIL	NOT	OF
OR	PACKED	PROCEDURE		PROGRAM
RECORD	REPEAT	SET	THEN	TO
TYPE	UNTIL	VAR	WHILE	WITH

A/2/3 **Speciální symboly**

+	-	*	/	=	<>	<
<=	>=	>	(	)	[	]
{	}	(*	*)	^	:=	..
,	;	:	'	.		

Poznámka : složené symboly pro porovnávání je třeba psát s použitím dvou jednoduchých symbolů.

A/2/3 **Předdefinované identifikátory**

Následující identifikátory je možno považovat za deklarované v bloku, který zahrnuje celý program a proto jsou dostupné v celém programu, pokud je programátor nepředefinuje v rámci vnitřního bloku. Bližší informace v kapitole 3.

```
CONST   MAXINT = 32767
TYPE    BOOLEAN = (FALSE, TRUE)
        CHAR (rozšířený soubor znaků ASCII)
        INTEGER = -MAXINT..MAXINT;
        REAL (podmnožina všech reálných čísel viz kapito-
```

la 2/3.)

```
PROCEDURE  WRITE; WRITELN; READ; READLN; PAGE; HALT; USER;
           POKE; INLINE; OUT; NEW; MARK; RELEASE; TIN; TOUT;
FUNCTION   ABS; SQR; ODD; RANDOM; ORD; SUCC; PRED; INCH;
           EOLN; PEEK; CHR; SQRT; ENTIER; ROUND; TRUNC;
           FRAC; SIN; COS; TAN; ARCTAN; EXP; LN; ADDR; SIZE;
           INP;
```

A/3/1/3 Reálná čísla

Je použit tvar (mantisa, exponent) podobný tvaru, který používá standartní vědecká notace - jenomže je použito binární místo desítkové soustavy. Například :

$2 = 2^{1}1\emptyset^{\emptyset}$ nebo $1.\emptyset_2 \cdot 2^{1}$

$1 = 1 \cdot 1\emptyset^{\emptyset}$ nebo $1.\emptyset_2 \cdot 2^{\emptyset}$

$-12.5 = -1.25 \cdot 1\emptyset^{1}$ nebo $-25 \cdot 2^{-1}$

$-11\emptyset\emptyset1_2 \cdot 2^{-1}$

$-1.1\emptyset\emptyset1_2 \cdot 2^{3}$ už normálně

$\emptyset.1 = 1.\emptyset \cdot 1\emptyset^{-1}$ nebo $1/1\emptyset = 1/1\emptyset1\emptyset_2 = \emptyset.1_{2/101_2}$

takže nyní musíme provést dlouhé dělení ve dvojkové soustavě.

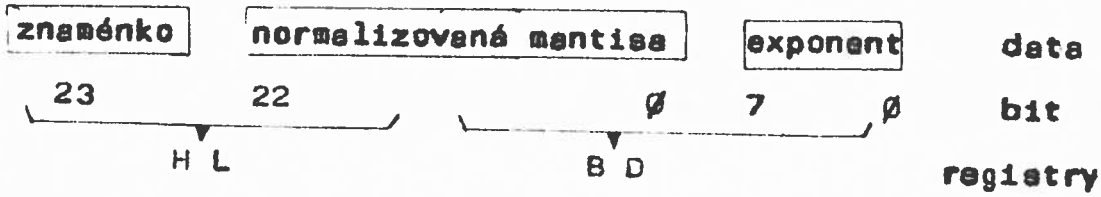
$\emptyset.1\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset : 1\emptyset1 = \emptyset.\emptyset\emptyset\emptyset11\emptyset\emptyset$

1ø1
11ø
1ø1
1øøø
1ø1

v tomto okamžiku vidíme, že zlomky se opakují

$\emptyset.1_2/1\emptyset1_2 = \emptyset.\emptyset\emptyset\emptyset11\emptyset\emptyset_2$ a výsledek je $1.1\emptyset\emptyset11\emptyset\emptyset \cdot 2^{-4}$

Takže, jak použijeme horní výsledky pro zobrazení těchto čísel v počítači? Tak zaprvé, rezervujeme 4 byty v paměti pro každé reálné číslo v následujícím formátu :



znaménko — znaménko mantisy; : = záporné, ø = kladné

normalizovaná mantisa — mantisa normalizovaná do tvaru 1.xxxxxx s horním bitem (bit 22) vždy 1, kromě případu kdy je nulové (HL = ø, DE = ø).

DOPLNĚK 3
VNITŘNÍ ZOBRAZENÍ A UKLÁDÁNÍ DAT

A/3/1 Vnitřní zobrazení dat

Následující rozbor upřesňuje, jakým způsobem HP4 zobrazuje data.

Informace o rozsahu potřebné paměti v jednotlivých případech, bude většinou pro programátora bez užitku (v kapitole 3/3/6/7 viz použití funkce SIZE); ostatní podrobnosti mohou potřebovat ti, kdož se pokouší kombinovat programy v Pascalu a strojovém kódu.

A/3/1/1 Celé čísla

Každé celé číslo zabírá v paměti 2byty v dvojkovém komplementárním tvaru. Například :

1 = ØØØ1

256 = Ø1ØØ

-256 = FFØØ

Standartní registr Z8Ø, používaný kompilátorem pro ukládání čísel je registr HL.

A/3/1/2 Znaky, booleanské výrazy a ostatní skaláry

Každý z těchto zabírá v paměti 1 byt v čistém, neindexovaném binárním tvaru.

Znaky: 8bitů, je použit rozšířený ASCII. (kód)

'E' = 45 '[' = 58

Booleanské výrazy :

ORD(TRUE) = 1 takže TRUE zastupuje jednotka

ORD(FALSE) = Ø takže FALSE zastupuje nula

Kompilátor používé standartní A registr Z8Ø.

exponent	exponent v dvojkovém komplementárním tvaru	
2	= Ø 1ØØØØØØ ØØØØØØØØ ØØØØØØØØ ØØØØØØØ1	(4Ø,ØØ,ØØ,Ø1)
1	= Ø 1ØØØØØØ ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ	(4Ø,ØØ,ØØ,ØØ)
-12.5	= 1 11ØØ1ØØ ØØØØØØØØ ØØØØØØØØ ØØØØØ11	(E4,ØØ,ØØ,Ø3)
Ø.1	= Ø 11ØØ11Ø Ø11ØØ11Ø Ø11ØØ11Ø 1111111Ø	(66,66,66,FC)

Uvědomíme-li si tedy, že HL a DE jsou používány pro uložení reálných čísel, abychom vnitřně zobrazili všechna výše uvedená čísla, tak bychom je měli ukládat do registrů následujícím způsobem :

2	LD HL,#4ØØØ	LD DE,#Ø1ØØ
1	LD HL,#4ØØØ	LD DE,#ØØØØ
-12.5	LD HL,#24ØØ	LD DE,#Ø3ØØ
Ø.1	LD HL,#6666	LD DE,#FC66

Poslední z příkladů ukazuje, před výpočty s binárními zlomky mohou být nepřesné; Ø.1 nemůže být přesně zobrazeno jako binární zlomek vzhledem k limitovanému počtu desetinných míst.

Poznámka : Reálná čísla jsou v paměti ukládána v pořadí :

E D L H.

A/3/1/4 Zápisy a pole

Zápisy zabírají takový rozsah paměti, jaký je součet jejich součástí.

Pole : jestliže n = počet prvků pole a s = dimenze každého prvku, pak počet bytů, které pole zabírá je : n*s.

Například :

ARRAY [1..1Ø] OF INTEGER potřebuje 1Ø*2 = 2Ø bytů

ARRAY [2..12,1..1Ø] OF CHAR má 11*1Ø=11Ø prvků a tak zabírá 11Ø bytů

A/3/1/5 Množiny

Množiny se ukládají jako bitové řetězce a tak jestliže základní typ má n prvků, tak počet použitých bytů je : (n-1) DIV 8 + 1 . Například :

SET OF CHAR vyžaduje (256-1)DIV 8 + 1 = 32 bytů

SET OF (blue,green,yellow) vyžaduje (3-1)DIV 8 + 1 to je 1 byt.

A/3/1/6 Ukazatelé

Ukazatelé zabírají 2 byty, obsahující adresu (ve formátu Intel t.j. dolní byt jako první) proměnné, na kterou ukazují.

A/3/2 Ukládání proměnných za běhu programu

Existují tři případy, kdy uživatel potřebuje informaci o tom jak jsou proměnné při běhu programu uloženy :

a) Globální proměnné - deklarovány v hlavním programovém bloku

b) Lokální proměnné - deklarovány ve vnitřním bloku

c) Parametry a výsledky - průběžně a pro procedury a funkce

Tyto jednotlivé případy jsou probrány dále a příklady jejich použití můžete najít v Doplňku 4.

Globální proměnné

Globální proměnné jsou řazeny od vrcholu programového zásobníku směrem dolů. Například : když programový zásobník je na #BØØØ a hlavní programové proměnné jsou :

```
VAR   i : INTEGER;
      ch : CHAR;
      x : REAL;
```

potom i (které zabírá 2 byty - viz předcházející odstavec A/3/1/1.) bude uloženo v paměti na #BØØØ-2 a #BØØØ-1 t.j. na adresách #AFFE a #AFFF.
ch(1byt) bude na #AFFE-1 t.j. #AFFD.
x (4byty) bude na #AFF9, #AFFA, #AFFB, #AFFC

Lokální proměnné

Lokální proměnné nejsou snadno přístupné přes zásobník, proto místo toho je IX registr nastaven na začátek každého vnitřního bloku (IX-4), takže ukazuje na začátek lokálních proměnných bloku. Například :

```
PROCEDURE test;
VAR i,j : INTEGER;
```

potom i (integer - 2 byty) bude umístěno na IX-4-1 a IX-4-2 t.j. IX-6 a IX-5.
j bude na IX-8 a IX-7

Parametry a výsledky

S hodnotami parametrů se zachází stejně jako s lokálními proměnnými a jako tyto proměnné, oč dříve je parametr deklarován o to vyšší adresu má v paměti. Na rozdíl od proměnných je nejnižší (na nejvyšší) adresa pevně dána IX+2.
Například :

```
PROCEDURE test(i : REAL; j : INTEGER);
```

pak j (zařazena napřed) je na IX+2 a IX+3
i je na IX+4, IX+5, IX+6 a IX+7

Proměnné jako parametry se obsluhují stejně jako hodnoty parametrů s výjimkou v tom, že jsou vždy přiřazeny 2 bytům a tyto 2 byty obsahují adresu proměnné. Například :

```
PROCEDURE test (i : INTEGER; VAR x : REAL);
```

pak odkaz na x je umístěn na IX+2 a IX+3 ; tato místa

paměti obsahují adresu, kde je uloženo x.

Hodnota i je na IX+4 a IX+5

VÝSLEDKY funkcí jsou umístěny nad prvním parametrem v paměti. Například :

```
FUNCTION test (i : INTEGER) : REAL;
```

potom i je na IX+2 aIX+3 a pro výsledek je rezervováno místo IX+4, IX+5, IX+6 a IX+7.

DOPLNĚK 4

PŘÍKLADY PROGRAMŮ HP4T

Máte-li jakékoliv pochybnosti o tom jak programovat v HISOFT PASCALu HP4T, tak byste si měl pozorně prostudovat následující programy.

Následující program ilustruje použití TIN a TOUT. Program vytváří velmi jednoduchý telefonní seznam na pásce a potom jej přečítá zpět. Měl byste napsat jakékoliv požadované vyhledávání.

```
PROGRAM TAPE;
CONST
  Size=10;              (Všimněte si, že"Size" je napsáno velkým
TYPE                     a malými písmeny - nikoliv "SIZE")
  Entry = RECORD
            Name : ARRAY [1..10]  OF CHAR;
            Number : ARRAY [1..10]  OF CHAR
          END;
VAR
  Directory : ARRAY [1..Size]  OF Entry;
  I : INTEGER;
BEGIN
  FOR I:= 1 TO Size DO
  BEGIN                    WITH Directory [I] DO
                           BEGIN
    WRITE("Name please") ;
    READLN;
    READ(Name);
    WRITELN;
    WRITE("Number please")
    READLN;
```

```
      READ(Number);
      WRITELN
    END
  END;
        (uložení seznamu na pásku pomocí ...)
TOUT("Director",ADDR(Directory) ,SIZE(Directory))
        (čtení pole zpět se provede následovně ...)
TIN("Director",ADDR(Directory))
        (a nyní můžete seznam zpracovávat podle přání)
END.
```

Program zapsanou řádku seřadí v obráceném pořadí, předvádí použití ukazatelů, zápisů, MARK a RELEASE.

```
10 PROGRAM ReverzeLine;
20 TYPE elem=RECORD          (Vytvoří strukturu zřetězeného
                                          seznamu)
30             next:↑elem;
40              ch: CHAR
50            END;
60       link=↑elem;
70 VAR prev,cur,heap: link;    (Všechny ukazatele pro "elem")
80  BEGIN
90  REPEAT                    (Dělej to mnohokrát)
100    MARK(heap);             (Urči vrchol zásobníku "heap")
110    prev:=NIL;          (Zde ještě není žádná proměnná)
120    WHILE NOT EOLN DO
130       BEGIN
140         NEW(cur);    (Vytváří nový dynamický zápis)
150         READ(cur↑.ch); (A přiřazuje jeho poli jeden
160                              znak ze souboru.)
170         cur↑.next:=prev;  (Tento ukazatel pole adresuje
180         prev:=cur;             předešlý záznam.)
```

```
19Ø        END;
    (Vytisknutí převrácených řádek po řádcích, v záznamu jsou
     obráceně.)
2ØØ    cur:=prev;
21Ø    WHILE cur <> NIL DO    (NIL je první.)
22Ø      BEGIN
23Ø        WRITE(cur .ch);    (Tiskni toto pole t.j. znak.)
24Ø        cur:=cur .next     (Adresa předcházejícího pole.)
25Ø      END;
26Ø    WRTELN;
27Ø    RELEASE(heap);         (zruš dynamické proměnné.)
28Ø    READLN                 (čekej na následující řádku)
29Ø  UNTIL FALSE              (použij EDIT k ukončení
3ØØ END.
```

Následující program vám ukáže použití rekurze, počítá faktoriál čísla vloženého z klávesnice.

1. použitím rekurzivního algoritmu
2. metodou iterace

```
1Ø PROGRAM FACTOR;
2Ø TYPE
3Ø   POSINT = Ø..MAXINT;
4Ø VAR
5Ø   METHOD : CHAR;
6Ø   NUMBER : POSINT;
7Ø FUNCTION RFAC(N : POSINT) : INTEGER;  (rekursivní algorit.)
8Ø  VAR F : POSINT;
9Ø  BEGIN
1ØØ   IF N>1 THEN F:= N * RFAC(N-1)  (RFAC se volá N krát)
11Ø           ELSE F:=1;
12Ø    RFAC:= F
```

```
13Ø  END;
14Ø FUNCTION IFAC (N : POSINT) : ITEGER; (Iterace)
15Ø   VAR I,F : POSINT;
16Ø   BEGIN
17Ø    F :=1;
18Ø    FOR I := 2 TO N DO F := F * I;  (jednoduchá smyčka)
19Ø    IFAC := F
2ØØ   END;
21Ø BEGIN
22Ø   REPEAT
23Ø    WRITE('Zvolte metodu (I nebo R) a číslo');
24Ø    READLN;
25Ø    READ(MATHOD,NUMBER);
26Ø    IF METHOD = 'R'
27Ø          THEN WRITELN(NUMBER,'! = ',RFAC(NUMBER))
28Ø          ELSE WRITELN(NUMBER,'! = ',IFAC(NUMBER))
29Ø   UNTIL NUMBER = Ø
3ØØ END.
```

Další program Vám ukáže jeden "trik", jak měnit proměnné Pascalu použitím strojového kódu.
Předvedeme Vám použití PEEK, POKE, ADDR a INLINE.

```
1Ø PROGRAM divmult2;
2Ø VAR r:REAL;
3Ø FUNCTION divby2(x:REAL):REAL;      (funkce dělení 2 ...
4Ø VAR i:INTEGER;                      .. rychlá)
5Ø BEGIN
6Ø  i := ADDR(x)+1;
7Ø  POKE(i,PRED(PEEK(i,CHAR)));  (dekrementuje exponent x
                                    viz doplněk 3/1/3.)
8Ø  divby2:=x
```

TZ 721066 82

```
 9Ø END;
1ØØ FUNCTION multby2(x:REAL):REAL;  (rychlá funkce násobení
11Ø BEGIN                                        2)
12Ø  INLINE($DD,$34,3;        (INC (IX+3) - exponentu x.)
13Ø  multby2:=x
14Ø END;
15Ø BEGIN
16Ø  REPEAT
17Ø   WRITE('Vložte číslo r');
18Ø   READ(r);                          (zde není nutné READLN viz
                                              kapitola 3/3/1/4.)
19Ø   WRITELN('r deleno 2 je ', divby2(r):7:2);
2ØØ   WRITELN('r násobeno 2 je ', multby2(r):7:2)
21Ø  UNTIL r=Ø
22Ø END.
```

L I T E R A T U R A

K.Jensen, N. Wirth

PASCAL USER MANUAL AND REPORT Springen-Verlag 1975

W.Findlay, D. A. Watt

PASCAL AN INTRODUCTION TO SYSTEMATIC PROGRAMING

Pitman Publishing 1978.

J. Tiberghien

THE PASCAL HANDBOOK SYBEX 1981

J. Welsh, J. Elder

INTRODUCTION TO PASCAL

První a třetí kniha je určena specialistům, zatímco druhá a čtvrtá jsou určeny pro začátečníky a jsou úvodem do jazyka.