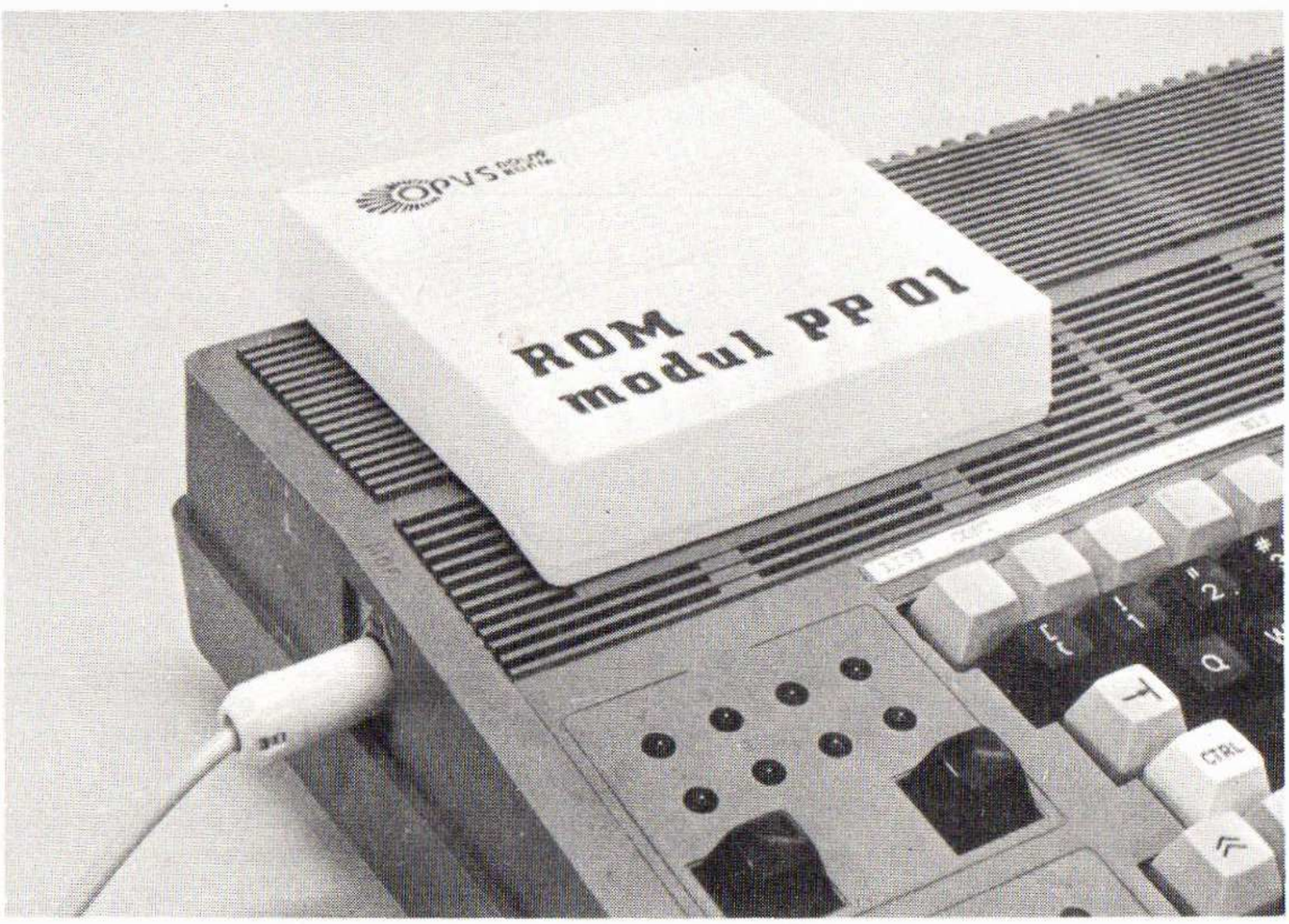
OKRESNÝ PODNIK VÝROBY A SLUŽIEB
DOLNÝ KUBÍN

ROM MODUL PASCAL

pre personálny počítač SMEP
PP 01



Príručka používateľa

OPVS DOLNÝ KUBÍN

Dokumentace systému Pascal pro mikropočítač PP-01, obsažená v tomto svazku, se skládá z následujících částí:

Příručka programátora
Příručka operátora
Popis editoru
Příloha A: Tabulka příkazů řídicího systému
Příloha B: Tabulka příkazů editoru
Příloha C: Odlišné varianty Systému Pascal

Dokumentace platí pro verzi 2.4

SYSTÉM PASCAL NA PP-01

Příručka programátora

Obsah:

1. Úvod

Jazyk Pascal je dnes, více než 10 let po svém zrodu, uznávaným prostředkem pro výuku programování. Často je používán i v praktických aplikacích. Dnes již má schválenou mezinárodní normu a jeho dialekty se liší relativně málo. Předkládaná implementace vychází z normy ISO 7185 (viz (1)).

Tato příručka není učebnicí Pascalu. Specifikuje vlastnosti verze jazyka Pascal implementované na mikropočítači PP-01. Popisuje

- implementací definované rysy jazyka,
- některé implementačně závislé rysy,
- vlastnosti jazyka, které se často v různých implementacích liší a mohly by být nejasné,
- jazyková omezení a rozšíření.

Dále příručka obsahuje vysvětlení chyb, které mohou nastat při překladu nebo běhu programu.

Konkrétní způsob práce s počítačem v systému Pascal je popsán v příručce operátora, na niž se budeme odvolávat zkratkou PO.

Odstavcům příručky, označeným „!", je vhodné věnovat zvýšenou pozornost. Značkou „§" jsou označeny odstavce, jejichž studium lze odložit až na druhé čtení příručky. Některé z nich mohou být při prvním seznámení s Pascalem hůře pochopitelné.

2. Literatura o Pascalu

Pro seznámení s jazykem Pascal doporučujeme učebnici (1). V (2) lze nalézt dobře srozumitelný výklad podmnožiny Pascalu. Obě tyto publikace by měly být snadno dostupné, což nelze říci o výborné knize (3). Dále je programování v Pascalu vyloženo ve skriptech (4) a (5).

3. První kroky

Tato kapitola je určena začátečníkům.

Po napsání programu v jazyku Pascal a po dobrém promyšlení jeho funkce lze zasednout k počítači.

Práce se systémem Pascal se obvykle skládá z několika po sobě následujících fází:

1. Editování programu v Pascalu
2. Překlad programu
3. Provádění přeloženého programu

V 1. fázi pracujeme s **editorem.** Vytváříme nebo opravujeme tzv. **zdrojový program,** tj. zápis programu v Pascalu.

Ve 2. fázi se zdrojový program přeloží do vnitřní formy nazývané **kód.** Během překladu se na obrazovce objevuje **protokol o překladu,** který obsahuje mimo jiné zdrojový program a informace o **syntaktických chybách.**

Je-li překlad bez chyby, lze přistoupit k 3. fázi. Je to provádění programu neboli **běh programu.** Během něj může dojít k **běhovým chybám.**

Při výskytu chyb v 2. nebo 3. fázi nebo při nesprávné funkci programu je třeba se vrátit k 1. fázi a program opravit.

Zdrojový program nebo kód lze nahrát na magnetofonovou kazetu. To nám umožní pozdějším nahráním z kazety nahradit opětovnou editaci nebo překlad.

4. Vlastnosti jazyka

4.1. Lexikální úroveň programu

4.1.1. ! Identifikátory a vyhražená slova

Pro zlepšení čitelnosti lze dovnitř identifikátorů vkládat znak „_" (podtrhávátko). Překladač tento znak v identifikátorech ignoruje. Podtrhávátko se nesmí vyskytnou uvnitř vyhražených slov.

V zápisu identifikátoru nebo vyhraženého slova má význam pouze prvních 6 znaků.

Za písmeno se považuje i znak měny.

Příklad:

Je_Tam a JETAM jsou přípustné zápisy stejného identifikátoru

GO_TO je nepřípustný symbol

NAPETI1 a NAPETI2 jsou stejné identifikátory (shodují se v prvních šesti znacích)

PROGRAMATOR je vyhražené slovo PROGRAM

4.1.2. Alternativní reprezentace

Ve funkci komentářových závorek lze použít i dvojznaků „(*" a „*)". Hranaté závorky lze nahradit „(." a „.)". Šipka má dvojí reprezentaci: „zavináč" (hexadecimálně 40H) a „oblouček" (hexadecimálně 5EH).

4.1.3. § Konstanta typu real

Část konstanty typu real uvedená před desetinnou tečkou musí být v absolutní hodnotě menší než 32768.

Lze tedy psát: 12345.6789123E13
ale nikoliv: 123456.789123E12

4.1.4. § Konstanta typu integer

Konstanty typu integer lze zapisovat i v hexadecimálním tvaru s dodržením těchto pravidel:
– prvním znakem zápisu musí být dekadická číslice;
– za posledním znakem napíšeme znak ‚H'.
Konstanta 8000H nepatří do typu INTEGER, lze ji použít pouze jako adresu. Konstanty větší než 8000H mají zápornou hodnotu (viz odst. 4.3.1.).

Příklad:
0A2H je zápis konstanty 162,
0FFFEH je zápis konstanty −2.

4.2. Kvantitativní omezení

4.2.1. §
V definici výčtového typu lze uvést nejvýše 256 konstant.
4.2.2. §
Lze použít nejvýše 6 úrovní statického vnoření procedur. (T.j. hlavní program a 5 vnořených procedur.) Dynamické vnořování je omezeno jen velikostí paměti.
4.2.3. §
Součet počtu procedur, počtu funkcí a počtu návěští, na něž vedou skoky ven z bloku, nemůže v celém programu přesáhnout 255.

4.3. Datové typy

4.3.1. ! Typ integer

Hodnota MAXINT je 32767, hodnoty typu integer musí tedy ležet v intervalu −32767 až 32767. Hodnoty jsou zobrazovány ve dvou bytech v doplňkovém kódu.

Při programování je nutné mít na paměti, že výsledky a mezivýsledky operací nad typem integer v absolutní hodnotě nesmějí přesáhnout hodnotu MAXINT.

4.3.2. § Typ real

Hodnoty typu real jsou zobrazovány ve čtyřech bytech. Nejmenší kladné zobrazitelné číslo je přibližně 1.18E−38, největší je přibližně 3.4E+37. Přesnost zobrazení je zhruba 6 desetinných míst.

4.3.3. Typ char

Znak se zobrazuje v 1 byte.

Uspořádání znakové množiny a ordinální čísla znaků jsou určeny kódem ASCII. Jako oddělovač řádek se používá dvojice znaků CR a LF (hexadecimálně 0DH a 0AH). Ty se při čtení z textového souboru jeví jako jedna mezera.

4.3.4. Typ boolean

Hodnoty typu boolean se zobrazují v 1 byte. Ordinální hodnota FALSE je 0, TRUE 1.

4.3.5. § Typy interval

Reprezentace je shodná s reprezentací hostitelského typu.

4.3.6. § Výčtové typy

Hodnoty se zobrazují v 1 byte.

4.3.7. Typy ukazatel

Hodnoty ukazatelů se zobrazují jako adresa ve 2 bytech.

Standardní procedury NEW a DISPOSE smějí mít právě jeden parametr – není tedy možné šetření pamětí vyznačením hodnoty rozlišovací proměnné v typu variantní záznam.

Procedura DISPOSE je implementována v plné síle – včetně zcelování paměti.

Mimo to jsou k dispozici standardní procedury MARK a RELEASE. Mají jediný argument libovolného typu ukazatel.

Zavoláním prodecury MARK(P) se proměnné P přiřadí adresa aktuálního vrcholu haldy. Nechť v dalším průběhu výpočtu halda vzroste. Zavoláním RELEASE(P) se uvolní část haldy nad místem, na něž ukazuje P.

Přístup k proměnné alokované v oblasti uvolněné pomocí RELEASE může (ale nemusí) vést k chybě při výpočtu.

Nelze doporučit neuvážené kombinování použití procedury DISPOSE s procedurami MARK a RELEASE. Systém Pascal by sice pracoval správně, ale špatně by se odhalovaly uživatelské chyby.

4.3.8. ! Typy množina

Hodnota typu množina se zobrazuje v 32 bytech paměti.

Typ množina lze vytvořit jen z takového typu, jehož hodnoty mají ordinální čísla v intervalu 0 až 255.

Lze tedy psát:

SET OF 0..250
SET OF BOOLEAN
SET OF (SRDCE, KRIZE, KARA, PIKY)
SET OF CHAR

ale nelze:

SET OF 1..1000
SET OF INTEGER

4.3.9. § Typy pole

Schéma konformního pole není implementováno. Prvky pole jsou při uložení do paměti uspořádány lexikograficky podle indexů.

4.3.10. Typy soubor

Proměnná typu soubor vyžaduje 5 bytů + paměť pro uložení jedné položky souboru.

Implementace připouští použití pouze externích souborů. Nelze tedy deklarovat soubor lokální v hlavním programu nebo proceduře, soubor nemůže být součástí strukturovaného typu a nemůže být dynamicky alokován.

Další informace o práci se soubory jsou v odstavci 4.4.

4.3.11. Typy záznam

Paměťové nároky typu záznam se rovnají součtu velikosti pevné části a největší větve variantní části.

Položky záznamu se ukládají za sebou v pořadí deklarace. Výjimku tvoří případ, kdy uvádíme několik

jmen položek stejného typu oddělených čárkami. Tyto položky se pak uloží v obráceném pořadí.

Příklad:
Položky záznamu:
RECORD A, B : REAL; C : INTEGER END
se ukládají v pořadí B, A, C. Záznam zabírá 10 bytů.

4.3.12. § Pakování typů

Definice strukturovaného typu může být prefixována vyhraženým slovem PACKED (typ může být „pakován"). Nemá to však žádný vliv na zobrazení hodnot typů ani na kompatibilitu typů. Lze tedy dosazovat pakované řetězce do nepakovaných; prvek pakované struktury smí být skutečným parametrem procedury volaným referencí. Standardní procedury PACK a UNPACK jsou zbytečné a nejsou implementovány.

4.4. Vstup a výstup

4.4.1. Standardní výstup

Při výstupu hodnot pomoci procedur WRITE a WRITELN lze explicitně zadat počet znaků, který má vystoupit do textového souboru. Implicitní hodnoty pro jednotlivé typy vystupujících dat udává následující tabulka:

Typ	počet znaků
INTEGER	8
REAL	16
CHAR	1
BOOLEAN	4 pro TRUE, 5 pro FALSE
řetězec	délka řetězce

4.4.2. § Ovládací programy souborů

Veškerá komunikace běžícího programu s okolím probíhá prostřednictvím souborů (výjimka viz odst. 4.7.3.). Každý soubor je čten nebo zapisován na jedno vnější zařízení prostřednictvím ovládacího programu.

Omezené technické prostředky PP-01 a absence operačního systému omezují možnosti práce se soubory v Pascalu. Proto:

1. Lze používat pouze externí soubory (viz odst. 4.3.10.)
2. Každý soubor smí být procedurou RESET nebo REWRITE otevřen nejvýše jednou. Standardní soubory INPUT a OUTPUT jsou otevírány systémem. Explicitní otevírání standardních souborů (RESET(INPUT), REWRITE(OUTPUT)) nebo opakované otevření ostatních souborů není dovoleno. Speciálně tedy nelze jediným programem vytvořit soubor a zpětně jej přečíst.
3. Zda je soubor vstupní nebo výstupní a na jakém technickém zařízení je realizován, je dáno ovládacím programem použitým pro soubor. Ovládací program přenáší jeden byte dat mezi souborem a programem.

Při otevírání souboru pro čtení se do vyrovnávací paměti čte první položka souboru. Výjimkou je soubor INPUT, u něhož se namísto čtení do vyrovnávací paměti zapíše mezera a soubor je ve stavu EOLN (tj. EOLN(soubor)=TRUE). Uživatel by měl počítat s touto mezerou navíc.

V systému Pascal jsou zabudovány ovládací programy pro následující zařízení a způsoby komunikace:

A. Ovládací program pro výstup na obrazovku
B. Ovládací program pro řádkový vstup z klávesnice
C. Ovládací program pro výstup na tiskárnu

D. Ovládací program pro snímání stavu klávesnice

Tyto ovládací programy používáme pro textové soubory.

Při čtení ze souboru ovládaného programem B se na obrazovce objeví dvojtečka a užívatel napíše řádek ukončený klávesou CR. Když program přečte zadaný řádek včetně oddělovače (např. procedurou READLN), vyžádá se vstup dalšího řádku. Konec souboru (EOF) lze zadat klávesou ADR.

Při čtení ze souboru řízeného ovládacím programem D se otestuje, zda je stisknuta nějaká klávesa. Pokud ano, přečte se její ASCII kód, v opačném případě se přečte hodnota 128. Tento ovládací program rozlišuje mezi velkými a malými písmeny vstupujícími z klávesnice, malá písmena ovšem nemohou vystoupit na obrazovku.

4. *Ovládací programy A a C lze použít jen pro výstupní soubory (otevírané pro zápis), ovládací programy B a D jen pro vstupní soubory (otevírané pro čtení).*
5. *Každému souboru je přiřazena adresa vchodu do jednoho ovládacího programu. Implicitně je to pro soubor INPUT ovládací program B, pro soubor OUTPUT ovládací program A, pro ostatní soubory uvedené v hlavičce programu ovládací program C.*
6. *Adresu jiného než implicitního ovládacího programu lze souboru přiřadit v hlavičce programu. Za identifikátorem souboru uvedeme dvojtečku a adresu ovládacího programu.*

Adresy ovládacích programů:

ovládací program	adresa	hexadecimálně
A: výstup obrazovka	16	10H
B: vstup klávesnice	59984	EA50H
C: výstup tiskárna	64	40H
D: stav klávesnice	60019	EA73H

Jiný způsob zadání adres ovládacích programů je v PO odst. 4.3.

Příklady hlaviček programu:

PROGRAM NORMALNI (INPUT, OUTPUT)
- soubor INPUT se čte po řádcích z klávesnice
- soubor OUTPUT vystupuje na obrazovku

PROGRAM S _ TISKARNOU (INPUT, OUTPUT, TISK)
- soubor INPUT se čte po řádcích z klávesnice
- soubor OUTPUT vystupuje na obrazovku
- soubor TISK vystupuje na tiskárnu

PROGRAM SPECIAL (OUTPUT:64, KONZOLA:60019, INPUT, DALNOPIS:30000)
- soubor OUTPUT vystupuje na tiskárnu
- soubor KONZOLA čte stav klávesnice
- soubor INPUT se čte po řádcích z klávesnice
- soubor DALNOPIS je zpřístupněn ovládacím programem uloženým od adresy 30000

4.5. Výrazy

Při vyhodnocování výrazů je třeba dbát na to, aby výsledky ani mezivýsledky operací nepřekročily rozsah hodnot daného typu.

4.5.1. § Operátor umocňování

*Operátor umocňování se zapisuje dvojznakem „**". Má větší prioritu než operátory typu násobení, ale menší prioritu než negace. Prvním operandem umocňování musí být celé číslo, druhým nezáporné celé číslo.*

*A**B**C je ekvivalentní (A**B)**C, nikoliv A**(B**C).*

4.5.2. § Změna typu

Změna typu je rozšířením Pascalu, kterého je třęba používat zvlášť obezřetně. Z hlediska syntaxe Pascalu jde o nový druh faktoru ve tvaru:

identifikátor_typu (faktor)

Vyhodnocení tohoto zápisu probíhá shodně s vyhodnocením faktoru uvedeného v něm. Na získanou hodnotu bude však překladač pohlížet tak, jako by byla toho typu, který je zadán identifikátorem. Změnu typu lze použít jen uvnitř těchto skupin typů:

1. *char, boolean, výčtový typ*
2. *integer, ukazatel*
3. *pole, záznam*

Při změně z typu skupiny 3. na 2. získáme adresu datové struktury a naopak.

Příklad:

I := INTEGER(P);	*kde P je ukazatel, I je typu INTEGER*
P := UKAZ(A);	*kde A je pole typu T, UKAZ typ ukazatele na T, P je typu UKAZ*

4.6. Příkazy

4.6.1. § Rozšíření syntaxe příkazu CASE

Uvnitř příkazu CASE lze vyznačit příkaz, který se má provést, pokud hodnota selektoru větvení se nerovná žádné z uvedených konstant. Před tímto příkazem uvedeme místo konstant (nebo vedle nich) vyhražené slovo ELSE. ELSE smí být v jednom příkazu CASE uvedeno nejvýše jednou.

Příklad:

```
CASE HODNOTA OF
  0: TYP := NULOVA;
1,2,3: TYP := MALA;
1000: TYP := VELKA;
ELSE: TYP := JINA
END
```

4.6.2. § Příkaz FOR

Při změně hodnoty řídící proměnné cyklu FOR uvnitř těla cyklu bude hlášena běhová chyba. Při vypnutí běhových kontrol tato změna stejně neovlivní počet průchodů cyklem.

4.6.3. § Příkaz GOTO

Skok dovnitř strukturovaného příkazu není vždy ohlášen jako chyba, ale chování programu bude nepředvídatelné. Viz také odst. 4.8.3.

4.7. Procedury a funkce

Viz také odst. 4.3.7. o procedurách na dynamické přidělování paměti.

4.7.1. Trigonometrické funkce

Argument trigonometrických funkcí je v radianech.

4.7.2. Procedurální a funkcionální parametry

Skutečným procedurálním nebo funkcionálním parametrem procedury nebo funkce nesmí být standardní procedura nebo funkce.

4.7.3. § Externí kompilace

Externí kompilace procedur nebo funkcí není na PP-01 možná. Direktiva EXTERNAL není implementována.

4.7.4. § Procedury pro práci se strojovou úrovní

Pro účely tohoto odstavce předpokládejme deklarace: TYPE BYTE = 0..255;
ADDRESS = INTEGER;

Má-li hodnota typu INTEGER význam adresy, pohlíží se na ní jako na dvoubytové celé číslo bez znaménka.

Systém Pascal obsahuje tyto nové standardní procedury a funkce:

FUNCTION PEEK (A:ADDRESS):BYTE;
— hodnotou je obsah paměťové buňky s adresou A
PROCEDURE POKE (A:ADDRESS, B:BYTE);
— zapíše do paměti na adresu A byte B
POZOR: Není-li na adrese A paměť RAM, dojde k úplnému zablokování systému počítače!
FUNCTION INP (PORT:BYTE):BYTE;
— jako hodnotu vrátí byte přečtený ze vstupní brány PORT
PROCEDURE OUT (PORT, B:BYTE);
— zapíše na výstupní bránu PORT byte B
PROCEDURE DISABLE;
— zakáže přerušení
PROCEDURE ENABLE;
— povolí přerušení
PROCEDURE CALL (START:ADRESA);
— provede podprogram ve strojovém kódu na adrese START. Podprogram musí být ukončen instrukcí RET. Procedura CALL smí mít ještě další parametry

typu INTEGER. Budou předány volanému podprogramu takto:

- uvede-li se jeden parametr, bude v dvojregistru BC i DE,
- uvedou-li se dva parametry, bude první v BC, druhý v DE,
- uvede-li se více parametrů, budou uloženy postupně na zásobníku (první nejhlouběji). Výjimkou jsou poslední dva parametry, předposlední bude v BC, poslední v DE.

FUNCTION FCALL (START:ADRESA) : INTEGER;

- má stejnou funkci a parametry jako CALL, navíc vrací hodnotu, která v okamžiku návratu z podprogramu byla v registru DE.

FUNCTION REF (PR:každý_typ) : ADRESA;

- funkce vrátí adresu proměnné PR. Pomoci konverze typů lze pak z adresy získat ukazatel na proměnnou.

4.7.5. Direktiva FORWARD

Direktiva FORWARD je implementována dle normy.

4.8. Zvláštní rysy implementace

4.8.1. § Rozsah platnosti identifikátoru

Oblast příslušející definici identifikátoru v deklarační části bloku začíná v místě definice, tedy není celým blokem. Díky tomu globální identifikátor může být lokálním identifikátorem zastíněn až po použití (globálního).

Příklad:

```
CONST I = 5;
PROCEDURE A;
TYPE ARRAY (.1..I.) OF CHAR;
VAR I : INTEGER;
```

— jde o správnou část programu, přitom v bloku procedury A má identifikátor I dva různé významy.

4.8.2. § Alokace proměnných

Paměť je lokálním proměnným procedur přidělena staticky. Při rekurzivním volání se lokální proměnné procedury odsunou na zásobník a po ukončení procedury se lokální proměnné kopírují zpět. V důsledku toho je v každém okamžiku výpočtu možný přístup jen k poslednímu exempláři lokálních proměnných rekurzivní procedury.

Toto omezení může vést k chybě, pokusíme-li se prostřednictvím referenčních parametrů rekurzivních procedur přistupovat ke starším exemplářům proměnných.

4.8.3. § Výskok z procedurálního parametru

Při výskoku příkazem GOTO z bloku procedury nebo funkce, která byla volána jako parametr, se skáče do poslední aktivní verze bloku. Lze sestrojit program (myšlenkově značně složitý), jehož chování díky tomu nebude odpovídat normě.

Příklad takového programu:

```
PROCEDURE DUMMY;
  BEGIN  END;
PROCEDURE A (PROCEDURE P);   FORWARD;
PROCEDURE B (PROCEDURE P);
```

```
    LABEL 1;
    PROCEDURE TOJEONA;
      BEGIN   GOTO 1   END;
  BEGIN
    P;   A(TOJEONA);   WRITELN(1);
  1 : END;
  PROCEDURE A;
    BEGIN   B(P);   WRITE(2)   END;
```

Při volání B(DUMMY) vystoupí do souboru OUTPUT čísla „2" a „1". Při chování podle normy by nic nevystoupilo.

4.8.4. Přidělování paměti proměnným

Paměť se jednotlivým zpracovávaným deklaracím proměnných přiděluje **od vyšších adres k nižším.** V rámci jedné deklarace se proměnným odděleným čárkami přidělí paměť **od nižších adres k vyšším** (viz příklad v kap. 6.). Množství přidělené paměti je určeno typem proměnné.

Parametrům volaným hodnotou se přidělí paměť stejně jako proměnným, parametrům volaným referencí se přidělí 2 byty paměti, procedurálním a funkcionálním parametrům 1 byte.

5. Řízení překladu

5.1. Přepínače

Práci překladače lze ovlivnit nastavením 8 přepínačů. Každý přepínač je označen jednim písmenem a jeho funkce může být aktivována nebo potlačena.

Přehled přepínačů je v tab. 5.1.

jméno	význam	implicitně
A	Generování testu při dosazování do proměnné typu interval	+
I	Generování testu při indexaci pole	+
P	Generování testu při dereferencování ukazatele	+
L	Vypisování protokolu o překladu (řádky s chybami nebo varováním se vypíšou vždy)	+
S	Hlášení použití rozšíření Pascalu oproti standardu	−
W	Varování při použití nebezpečné konstrukce	+
T	Tisk protokolu o překladu na tiskárně	−
V	Generování testu, zda hodnota proměnné není nedefinovaná	+

Přepínače A, I, V a P ovlivní množství testů prováděných při výpočtu programu a tedy i rychlost výpočtu. Přepínače L, S, W a T se týkají protokolu o překladu.

Implicitní nastavení přepínačů před začátkem překladu je v pravém sloupci tab. 5.1. Lze je změnit při spouštění překladu způsobem popsaným v PO odst. 4.2.

Nastavení přepínačů lze měnit kdekoliv v programu na začátku komentáře takovýmto zápisem:

(*$X+,Y− ... *)

kde X, Y jsou jména přepínačů, „+" způsobuje aktivaci a „−" potlačení funkce přepínače, $ je znak měny (ne paragraf).

5.2. Řízení tisku protokolu o překladu

Do zdrojového programu lze umístit řádky začínající znakem měny, které slouží k řízení tisku protokolu o překladu. Bezprostředně za znakem měny následuje jeden z těchto příkazů:

TITLE nadpis nahradí dosavadní nadpis tištěný v záhlaví každé stránky novým a přejde v protokolu na novou stránku.

CHANGE nadpis nahradí dosavadní nadpis tištěný v záhlaví každé stránky novým bez přechodu na novou stránku.

EJECT přechod na novou stránku v protokolu.

SPACE N do protokolu vystoupí N prázdných řádek, N je číslice.

UNTITLE potlačí všechny kompilátorem generované přechody na novou stránku, dokud se nenarazí na příkaz TITLE nebo EJECT.

Nadpis je dlouhý 24 znaků, případné další znaky se ignorují. Počáteční hodnota nadpisu je 24 mezer (je prázdný). Pouhé zrušení nadpisu při zachování přechodů na novou stránku se provede příkazem CHANGE s prázdným nadpisem.

6. Protokol o překladu

Strukturu protokolu o překladu ukážeme na následujícím úseku protokolu:

```
15 2F82 — B PROCEDURE INSERT (A: INTEGER);
16 2F80 —    VAR R: REAL;
17 2F7C —    X, Y: CHAR;
18 2F7A 0– B BEGIN
19 0B13 —    R: = A;
20 0B21 1–   IF A = 0 THEN BEGIN
```

V levém sloupci protokolu se nachází pořadové číslo zdrojové řádky. Započítávají se i řádky s příkazy pro řízení tisku protokolu (viz 5.2.).

Třetí sloupec protokolu obsahuje informace o úrovni vnoření příkazů. Při každém výskytu BEGIN, CASE, REPEAT na řádce se levý indikátor vytiskne a zvětší o 1. Při uzavření konstrukce se indikátor zmenší o 1 a vytiskne.

Písmeno v následujícím sloupci udává úroveň statického vnoření bloků. Tiskne se na začátku hlavičky procedury, u BEGIN a END vymezujících blok. Pro hlavní program se tiskne „A", pro proceduru v hlavním programu „B" atd.

Dále následuje přesný opis zdrojové řádky.

Význam druhého sloupce čísel se liší podle toho, zda na řádce vpravo jsou příkazy nebo deklarace. Je-li na řádce příkaz, číslo udává hexadecimální adresu, od níž se příkaz začal ukládat do paměti. Např. podmíněný příkaz v příkladu se uložil od adresy 0B21H. Je-li na řádce deklarace proměnných nebo parametrů procedury, číslo udává adresu o 1 větší než adresa posledního byte oblasti přidělené první proměnné.

Podle údajů o paměťových nárocích proměnných (4.3.) a podle konvencí v 4.8.4. lze nyní vypočítat, na jaké adrese je daná proměnná.

Ve výše uvedeném příkladu je:
parametr A na adrese 2F80H = 2F82H − 2,
proměnná R na adrese 2F7CH = 2F80H − 4,
proměnná Y na adrese 2F7BH = 2F7CH − 1,
proměnná X na adrese 2F7AH = 2F7BH − 1.

7. Chyby při překladu programu

Při překladu programu může zásadně dojít ke dvěma druhům chyb. První druh jsou chyby fatální, po nichž se vypíše zpráva o chybě a je ukončen překlad.

Druhý, podstatně častější druh chyb, jsou syntaktické chyby ve zdrojovém programu. Zprávy o těchto chybách se objeví v protokolu o překladu za místem detekce chyby. Číslo chyby dovolí určit, jakého prohřešku proti Pascalu se programátor dopustil.

Po syntaktické chybě se překladač snaží vzpamatovat a pokračovat v překladu. To se mu někdy dost dobře nepovede, a proto vzápětí hlásí i další, tzv. zavlečené chyby, které vznikly v důsledku chyb předchozích. Je-li v jednom místě programu hlášena skupina chyb, mívá největší význam první z nich.

Vysvětlení chyb:

Fatální chyby:

1 příliš dlouhý program, kód se nevejde do pracovní oblasti
2 více než 6 úrovní vnoření podprogramů není dovoleno
3 příliš mnoho procedur a funkcí
4 vnitřní chyba, překladač se nedokáže vzpamatovat z chyb uživatele
5 příliš rozsáhlé deklarace
6 deklarace příliš rozsáhlého typu záznam
7 dezintegrace překladače, proveď inicializaci
8 nedostatek paměti pro provedení překladu

Syntaktické chyby:

1 nepřípustný znak ve zdrojovém textu
2 vstupní řádka je příliš dlouhá
3 znak „_" se nesmí vyskytovat uvnitř vyhraženého slova
4 znakový řetězec nesmí přesáhnout řádku
5 prázdný znakový řetězec je nepřípustný
6 za identifikátorem konstanty nebo typu očekávám „="

7 příliš velká integerová konstanta nebo celá část reálné konstanty
8 konstanta začíná nepřípustným symbolem
9 použití lokálních souborů není na PP-01 možné
10 nepřípustný symbol
11 očekávám „,“ nebo „)“
12 tělo bloku musí začínat vyhraženým slovem „BEGIN“
13 očekávám „:“ a identifikátor typu funkce
14 očekávám identifikátor konstanty
15 skutečným parametrem volaným referencí musí být proměnná
16 očekávám identifikátor typu
17 faktor začíná nepřípustným symbolem
18 procedury READ a WRITE musí mít parametry
19 příliš málo skutečných parametrů
20 skutečných parametrů je příliš mnoho
21 skutečný procedurální/funkcionální parametr není kompatibilní s formálním parametrem
23 Identifikátory INPUT a OUTPUT se smí v hlavičce programu uvést nejvýše jednou
25 očekávám „..“
26 očekávám „,“ nebo „.)“
28 očekávam „PROGRAM“
32 očekávám tečku
45 očekávám levou závorku
46 očekávám čárku
47 očekávám celočíselnou konstantu
48 očekávám pravou závorku
49 očekávám „:=“
50 očekávám středník
51 očekávám dvojtečku
52 očekávám „THEN“
55 očekávám „END“
56 očekávám „UNTIL“

58 očekávám „OF"
59 očekávám „DO"
60 očekávám „TO" nebo „DOWNTO"
69 očekávám identifikátor
70 argumenty operací AND/OR musí být typu BOOLEAN
71 argumenty operací DIV/MOD musí být typu INTEGER
72 znaménko smí být jen před výrazem nebo konstantou typu INTEGER nebo REAL
73 negace smí být jen před faktorem typu boolean
74 ukazatelé smí být pouze v relacích rovno a nerovno
75 pro BOLEAN/CHAR/řetězec/výčtový typ nelze použít operací „+"/„–"/„*"/„/", pro množiny je nepřípustná operace „/"
76 argument musí být typu REAL nebo INTEGER
77 argument musí být typu INTEGER
78 argument musí být typu ukazatel
79 argument musí být typu soubor, argument EOLN/PAGE musí být typu text
80 procedury READLN/WRITELN lze použít pouze na textový soubor
81 nelze číst hodnotu identifikátoru funkce
82 z textového souboru lze číst pouze hodnoty typu CHAR/INTEGER/REAL
83 do textového souboru lze zapisovat pouze hodnoty typu CHAR/INTEGER/REAL/BOOLEAN/řetězec
84 zde musí být uveden výraz typu INTEGER
85 typ není ordinální
86 typ indexu musí být ordinální
87 obě meze v typu interval musí být stejného typu
88 typ výrazu v dosazovacím příkazu/parametru volaném hodnotou nebo část položky souboru

nesmí být typu soubor

89 ordinální čísla hodnot bázového typu musí ležet v intervalu 0 až 255

90 typ funkce nesmí být strukturovaný

91 očekávám identifikátor typu

92 typ druhé meze v definici typu interval je nepřípustný

93 nepřípustný symbol na začátku specifikace typu

94 výraz musí být typu BOOLEAN

95 typy prvků množiny nejsou kompatibilní

96 typy nekompatibilní nebo nekompatibilní vzhledem k dosazení

97 typy formálního a skutečného parametru volaného referencí nejsou shodné

98 skutečným procedurálním/funkcionálním parametrem musí být procedura nebo funkce

99 standardní procedura/funkce nesmí být skutečným parametrem

100 identifikátor není deklarován

101 návěští není deklarováno

102 dvojnásobná deklarace identifikátoru

103 dvojnásobná deklarace návěští

104 dvojí použití stejné konstanty v CASE

105 dvojí použití ELSE jako konstanty v CASE

106 dvojí definice návěští

107 dolní mez typu interval je větší než horní mez

109 skok dovnitř strukturovaného příkazu není dovolen

110 očekávám konstantu typu INTEGER

111 volání procedury nesmí být součástí výrazu

112 očekávám proměnnou

113 řídicí proměnná cyklu musí být lokální

114 rozlišovací konstanta v CASE musí být ordinálního typu

115 v příkazu CASE je nutno uvést alespoň jednu

větev

116 není to indentifikátor konstanty

117 proměnná není typu záznam

118 položka tohoto jména neexistuje

119 proměnná není typu pole

120 proměnná není typu ukazatel ani soubor

121 typ výrazu, kterým se indexuje, není kompatibilní s typem indexu

122 funkci lze přiřadit hodnotu jen v těle jejího bloku

123 identifikátoru standardní funkce nelze přiřadit hodnotu

124 parametrem musí být proměnná typu znakový řetězec

125 parametrem musí být proměnná typu INTEGER

126 v zápisu reálné konstanty musí po desetinné tečce následovat číslice

127 v exponentu reálné konstanty musí být alespoň jedna číslice

128 tato reálna konstanta je příliš velká nebo příliš malá

129 nevhodný druh identifikátoru

130 operandy umocňování musí být typu INTEGER

131 soubor INPUT nebyl uveden v hlavičce programu

132 soubor OUTPUT nebyl uveden v hlavičce programu

133 externí soubory musí být deklarovány (v hlavním programu)

134 výčtový typ nesmí obsahovat více než 256 konstant

140 příliš velký rozsah proměnných (více než 2**16-1 bytů)

141 příliš velký typ (datová struktura vyžaduje více než 2**16-1 bytů)

150 na této řádce jsou ještě další chyby

151 nějaké předsunuté deklaraci procedury/funkce

neodpovídá žádná identifikace
152 nedefinovaný identifikátor doménového typu
153 deklarované, ale nedefinované návěští
170 kolize kódu a proměnných, malý adresový prostor pro překládaný program

Kromě hlášení chyb překladač v některých případech vydá varování:
201 test na rovnost nebo nerovnost s operandy typu REAL není vhodný
202 v komentáři se vyskytl symbol „;" nebo „(*"

Použití rozšíření oproti normě Pascalu může být v závislosti na přepínači „S" hlášeno s čísly většími než 240:

245 zápis celočíselné konstanty v šestnáctkové soustavě
246 použití standardní procedury nebo funkce, která není součásti standardního Pascalu
247 návěští má více než 4 cifry
248 změna typu
250 operace umocňování

8. Chyby při běhu programu

Množství běhových kontrol prováděných v Pascalu může pomocí běhových chyb zabránit nedefinovanému průběhu výpočtu. Proto je rušení běhových kontrol (viz 5.1.) vhodné až u odladěných programů a při zvláštních požadavcích na rychlost výpočtu nebo délku kódu.

Běhová chyba způsobí ukončení výpočtu. Význam běhové chyby lze odvodit z čísla chyby. V závorkách jsou uvedeny příkazy, při nichž může chyba nastat.

K běhové chybě nedojde při pokusu o přístup k neaktivní větvi v typu záznam. Překladač také nekontroluje fyzickou existenci a připravenost ke komunikaci u nestandardních vnějších souborů a tiskárny.

Zvláštním způsobem jsou zpracovávány chyby 8, 9 a 30. Při jejich výskytu se neukončí běh programu, ale objeví se hlášení:

PRI CTENI CISLA CHYBA xx

pak se vypíše chybná část vstupní řádky a vyžádá se oprava chyby.

0 přerušení programu klávesou INT 3 – nejde o chybu
1 pokus o zápis do nestandardního souboru před provedením REWRITE (WRITE, WRITELN, PUT)
2 pokus o čtení z nestandardního souboru před provedením RESET (READ, READLN, GET)
3 pokus o čtení ze souboru po přečtení všech položek – při EOF (READ, READLN, GET)
4 překročení dolní meze hodnot jednobytového typu interval nebo výčtového typu (dosazování do proměnné typu interval, indexace)
5 překročení horní meze hodnot jednobytového typu interval nebo výčtového typu (dosazování do proměnné typu interval, indexace)
6 překročení dolní meze typu interval vytvořeného z typu INTEGER (dosazování do intervalu, indexace)
7 překročení horní meze typu interval vytvořeného z typu INTEGER (dosazování do intervalu, indexace)
8 na vstupu není číslice (READ, READLN pro argument typu INTEGER nebo REAL)
9 na vstupu je příliš velké číslo (READ, READLN)
10 počet znaků, které mají vystoupit do souboru, není

kladný (WRITE, WRITELN)

11 pokus o otevření souboru, který je již otevřen (RESET, REWRITE)

14 použití proměnné, které dosud nebyla přiřazena hodnota (použití hodnoty proměnné)

POZOR: Tato chyba může být také hlášena při některých zvláštních hodnotách proměnných. V takovém případě vypněte přepínač V (viz odst. 5.1.).

15 chybný parametr DISPOSE (DISPOSE)

16 vyčerpání dynamicky alokované paměti (NEW)

17 hodnota ukazatele je NIL nebo leží mimo haldu (přístup k identifikované proměnné, DISPOSE)

20 hodnotu typu REAL nelze převést na typ INTEGER – je příliš velká (TRUNC, ROUND)

21 podtečení při sčítání/odečítání pro typ REAL (+, –)

22 přetečení při sčítání/odečítání pro typ REAL (+, –)

23 podtečení při násobení/dělení pro typ REAL (*, /)

24 přetečení při násobení/dělení pro typ REAL (*, /)

25 dělení nulou (/)

30 příliš velké reálné číslo na vstupu (READ, READLN pro typ REAL)

32 hodnota skutečného parametru neleží v intervalu 0 až 255 (CHR, POKE, INP, OUT)

33 menší hodnota v tomto typu neexistuje (PRED)

34 větší hodnota v tomto typu neexistuje (SUCC)

35 hodnotou selektoru v CASE není označena žádná větev (příkaz CASE)

36 pokus změnit hodnotu řídící proměnné v těle cyklu (příkaz FOR)

47 hodnota nemůže být prvkem množiny (konstruktor množiny)

48 překročení mezí bázového typu (dosazování množin)

50 vyčerpání paměti (rekurzivní volání procedur a funkcí)

61 přetečení nebo podtečení v aritmetických operacích pro typ INTEGER (+, –, *, SQR)
62 exponent v celočíselném mocnění musí být nezáporný (**)
63 dělení celého čísla nulou (DIV, MOD)
64 druhý operand v operaci MOD je záporný (MOD)
80 příliš velký argument exponenciály (EXP)
81 argument logaritmu je nula (LN)
82 argument logaritmu je záporný (LN)
83 argument odmocniny je záporný (SQRT)
88 nepřípustný formát pro výstup reálné hodnoty (WRITE, WRITELN)

Po chybě se na obrazovce objeví nápis:

BEHOVA CHYBA,
TISK DUMPU NA TISKARNU? (A/N)

Stiskneme-li klávesu „A", vypíšou se na připojenou tiskárnu informace o situaci, v níž k chybě došlo. Stiskneme-li libovolnou jinou klávesu, vypisují se informace na obrazovku. Příklad takového výpisu následuje:

*** CHYBA *** 63
PROGRAM PRERUSEN NA ADRESE 0312
V BLOKU FIBO
LOKALNI PROMENNE (HEX):
79EF: 00
79F0: 00 00 7F

VOLANI NA ADRESE 033E V HLAVNIM
PROGRAMU
LOKALNI PROMENNE (HEX):
79F3: 41 D5 04 FE 01
79F8: 00 DC DA 50 EA 01 00 20

HALDA OD: 034D DO: 034D
ZASOBNIK OD: 79DF DO: 79EF

KONEC ZPRAV

Na první řádce chybového hlášení je číslo chyby 63. Z přehledu běhových chyb zjistíme, že došlo k dělení nulou v operaci MOD nebo DIV. Chyba nastala při provádění procedury nebo funkce FIBO na adrese 0312H (může nastat případ, že tato adresa nesouhlasí). V protokolu o překladu najdeme zdrojový řádek, který se přeložil do kódu uloženého na tuto adresu.

Dále následuje hexadecimální výpis hodnot lokálních proměnných procedury FIBO. Hodnoty DC zpravidla znamenají, že proměnné dosud nebyla přiřazena hodnota. Viz odst. 4.8.4. a kap. 6.

Chybové hlášení dále obsahuje informaci o celém pracovním řetězci, tj. o všech procedurách a funkcích aktivních v okamžiku výskytu chyby. Uvádí se vždy adresa, na níž byla z procedury vyvolána následující procedura a obsah paměti pro lokální proměnné. V našem případě byla procedura FIBO vyvolána přímo z hlavního programu na adrese 033EH, následují tedy proměnné deklarované v hlavním programu.

Hlášení je ukončeno informací o adresovém prostoru, v němž je halda a zásobník. Obsah těchto buněk paměti lze zjistit z řídicího systému příkazem MEMORY (viz PO 4.11.).

9. Závěr

Jedna programátorská poučka praví:

„Počítač pracuje podle Vašeho programu, nikoliv podle Vašeho přání."

Přejeme Vám mnoho úspěchů při práci s programovacím jazykem Pascal.

Autoři

Literatura

(1) Jinoch, J., Müller, K., Vogel, J.: Programování v jazyku Pascal. Praha, SNTL 1985.

(2) Wirth, N.: Systematické programovanie. Bratislava, ALFA, Praha, SNTL 1981.

(3) Wirth, N.: Algorithms + Data Structures = Programs. London, Prentice Hall 1976. (Slovenský překlad v tisku)

(4) Müller, K.: Programovací jazyky. (skriptum) Praha, ČVUT 1980.

(5) Ochranová, R.: Úvod do programování. (skriptum) Brno, UJEP 1982.

SYSTÉM PASCAL NA PP-01

Příručka operátora

Obsah:

1. Úvod

Přídavný paměťový modul „PP-01 Pascal" spolu se souborem nahraným na magnetofonové kazetě vytvářejí tzv. Systém Pascal na mikropočítači PP-01. Systém Pascal dovoluje používat vyšší programovací jazyk Pascal, čímž rozšiřuje standardní programové vybavení PP-01. Umožňuje zejména:

– vytvářet a upravovat programy v Pascalu,
– překládat je a spouštět,
– nahrávat na kazetu a z kazety zdrojové programy a přeložený kód.

Mimo to lze ze systému Pascal provádět nezákladnější příkazy monitoru.

Tato příručka popisuje práci se systémem Pascal. Nejjednodušší způsob, jak systém zvládnout, je posadit se k počítači a vyzkoušet si v praxi obsah kapitol 2. až 4. Zbývající kapitoly rozšiřují popis o speciální možnosti práce s mikropočítačem. Při prvním čtení příručky je lze bez obav vynechat.

Jazyková stránka systému Pascal je popsána v příručce programátora, na niž budeme odkazovat zkratkou PP. Je žádoucí si ji alespoň zběžně prostudovat před psaním prvního programu v Pascalu.

Poslední příručka, která se vztahuje k Pascalu, se jmenuje „Systém Pascal – Editor". Popisuje způsob vytváření a opravování zdrojových programů.

2. Uvedení systému do chodu

Před zapnutím počítače ověříme, zda
- v levé části panelu počítače je zasunut paměťový modul „PP-01 Pascal",
- k počítači je připojen televizní přijímač a je vyladěn na 4. kanál,
- k počítači je připojen kazetový mgnetofon,
- síťové šňůry všech zařízení jsou připojeny na síť.

Připojení televize, magnetofonu a paměťového modulu je popsáno v příručce uživatele PP-01.

Pokud se skutečné chování počítače odchýlí od níže popisovaných reakcí, podívejte se na konec kapitoly (postup při chybách).

Postup uvádění do chodu:

==) Zapněte síťový vypínač PP-01.

Na obrazovce se objeví nápis: GBASIC Vn.m
READY

==) Zadejte příkaz ROM.

Zadáváme-li příkaz, píšeme na klávesnici počítače jako na psacím stroji. Na konci celého příkazu vždy stiskneme klávesu CR (zcela vpravo v hlavní části klávesnice).

Na obrazovce se objeví nápis: NAHREJ PREKLADAC
PRIPRAV KAZETU

==) Do kazetového magnetofonu zasuňte kazetu dodávanou v rámci systému Pascal. Přitom:
- kazeta musí být otočena stranou A směrem ven,
- kazeta musí být převinuta na začátek strany A.

==) Stiskněte libovolnou klávesu počítače a zapněte přehrávání. Probíhá-li čtení správně, bude trvat asi dvě a půl minuty, obrazovka bude rolovat nahoru.

Po bezchybném nahrání se na obrazovce objeví nápis: SYSTEM PASCAL, Vi.j
==) Vypněte magnetofon.

Na obrazovce se v levém sloupci objevilo rovnítko „=" a blikající čtvereček – kurzor. To znamená, že můžete psát příkazy řídicího systému (viz další kapitoly).

Postup při chybách:

Pokud se po zapnutí počítače nepřihlásil GBASIC, postupujeme takto:

- je-li obrazovka televize stejnoměrně tmavá (i po přidání jasu), je závada na televizi. Zkontrolujeme zejména její připojení do sítě, zapnutí síťového vypínače, pojistky.
- nerozsvítila-li se v levé části panelu počítače kontrolka označená RUN, je chyba v počítači. Zkontrolujeme zapojení počítače do sítě, pojistku.
- je-li na obrazovce televize jen šum (sněžení), který neovlivnilo zapnutí počítače, zkontrolujeme kabel spojující počítač s televizí, správnou předvolbu kanálu resp. vyladění kanálu.
- reaguje-li televize na zapnutí počítače, ale neobjeví-li se přesný obraz, doladíme kanál.

Pokud po zadání příkazu ROM nenastane správná reakce (např. dojde k chybě 21), zkontrolujeme, zda je modul správně zasunut.

Pokud není úspěšné nahrávání překladače:

- zkontrolujeme správné propojení počítače a magnetofonu,
- zkontrolujeme, zda je v magnetofonu správná kazeta, ze strany „A" a od začátku,
- zopakujeme nahrávání.

3. Základy systému Pascal

Operátor, pracující se systémem Pascal, je ve styku s tzv. řídicím systémem.

Řídicí systém umístí na začátek nové řádky znak „=". Tento znak napovídá uživateli, že komunikuje s řídicím systémem a že může napsat příkaz.

Při psaní příkazu se jednotlivé znaky objevují v místě, kde byl kurzor (blikající čtvereček) a kurzor se posunuje doprava. Napíšeme-li chybný znak, lze jej zrušit stisknutím klávesy označené DEL nebo šipkou doleva. Kurzor se posune zpět na místo zrušeného znaku. Takto lze rušit i více znaků na jedné řádce. Celou řádku najednou lze zrušit stisknutím CTRL X (podržíme klávesu CTRL a stiskneme klávesu X).

Po napsání celého příkazu vždy stiskneme klávesu CR. V tom okamžiku se začne provádět akce zadaná příkazem. Během jejího provádění se na obrazovce mohou objevovat různé zprávy. Ukončení akce poznáme podle toho, že se vrátíme do řídicího systému a na začátku nové řádky se objeví „=" a kurzor.

Chceme-li přerušit provádění nějaké akce vyvolané z řídicího systému (např. proto, že neprobíhá podle našich představ), stačí stisknout klávesu INT 0 – na panelu je druhá zleva, v rámečku s kontrolkami přerušení.

Vypisování textu na obrazovku může být dosti rychlé. Chceme-li si text dobře prohlédnout než zmizí, stiskneme CTRL S. Vypisování se zastaví, lze v něm pokračovat stisknutím CTRL Q. Pozastavování výpisu je vhodné zejména při překladu programu.

Zadáte-li příkaz INITIALIZE, GO, BASIC nebo EDITOR s parametrem N, na obrazovce se objeví nápis:

JSES SI JIST?

Chápejme to jako varování před destruktivními násled-

ky těchto příkazů. Příkaz se provede, stiskneme-li klávesy A (ano), H (hej) nebo Y (yes). Stisknutí libovolné jiné klávesy příkaz zruší.

Příkaz nemůže být delší než řádka na obrazovce.

4. Příkazy řídicího systému

Pro příkazy, kterými z řídicího systému vyvoláváme provádění zvolených akcí, platí tato pravidla:

A. Příkaz vždy začíná jménem – písmenem označujícím příkaz.

B. Bezprostředně za jménem mohou následovat tzv. argumenty příkazu.

C. Je-li argumentem příkazu číslo, zadává se hexadecimálně (v šestnáctkové soustavě jsou číslice 10 až 15 nahrazeny písmeny A až F). Napíšeme-li více než 4 cifry, platí vždy jen poslední 4. Tak např.:
003F8 3F8 52A03F8
jsou ekvivalentní zápisy čísla 3F8H, dekadicky 1016.

D. Mezery se používají pouze k oddělení bezprostředně po sobě následujících čísel (v rozšířených příkazech LOAD a SAVE). Mimo to se v zápisu příkazu mezery nesmějí vyskytovat.

Příkazy popsané v následujících odstavcích jsou seřazeny zhruba podle důležitosti a frekvence použití. Přehledná tabulka příkazů je v příloze A.

4.1. Příkaz EDITOR – E

Příkaz způsobí vyvolání editoru – modulu, umožňujícího vytváření nebo opravování zdrojových programů v Pascalu.

Práce s editorem je popsána v příručce „Systém Pascal – Editor".

Jméno příkazu je E. Jediný přípustný argument je písmeno N. Uvedeme-li tento argument, smaže se starý zdrojový program a můžeme editovat nový. Neuvedeme-li N, budeme opravovat starý zdrojový program.
Příklad:
=E – editace programu uschovaného v paměti
=EN – smazání programu a editace nového programu

Zadáme-li omylem EN místo E, lze zdrojový program zachránit způsobem popsaným v kapitole 11.

4.2. Příkaz COMPILE – C

Příkaz vyvolá překlad zdrojového programu. Před spuštěním překladu musí být zdrojový program vytvořen editorem nebo nahrán příkazem READ.

Pomoci argumentů příkazu lze provést počáteční nastavení přepínačů překladače. Význam přepínačů je popsán v PP odst. 5.1.

Aktivaci funkce přepínače vyznačíme argumentem „X+", potlačení „X-", kde X je jméno přepínače. Nastavení přepínačů oddělujeme mezi sebou čárkami.

Příklad:
=C – spuštění překladu, počáteční nastavení přepínačů je implicitní (popsáno v PP tab. 5.1.)
=CS+,L– – spuštění překladu, při němž se budou hlásit rozšíření proti standardu a nebude vytvářen protokol.

Po zahájení překladu se na obrazovce přihlásí překladač a začne se vypisovat protokol. Pokud protokol nevystupuje na tiskárnu, je dobré si z něj opsat zprávy o chybách. Po ukončení překladu obdržíme zprávu o výsledku.

Další argumenty příkazu COMPILE jsou uvedeny v kap. 8.

4.3. Příkaz RUN – R

Příkaz RUN spustí běh přeloženého programu. Lze jej použít pouze tehdy, pokud byl dříve úspěšně a bezchybně přeložen zdrojový program nebo kód byl nahrán z kazety (příkazem LOAD).

Neuvedeme-li žádné argumenty, pak při běhu programu se výstup do souboru OUTPUT objeví na obrazovce a čtení ze souboru INPUT způsobí čtení po řádcích z klávesnice.

Chceme-li přerušit běh programu, stiskneme klávesu INT 0 nebo INT 3. Klávesa INT 3 vyvolá běhovou chybu 0, díky čemuž obdržíme informace o místě, na němž byl program přerušen – takzvaný dump (viz PP kap. 8.).

Zbývající část odstavce 4.3. se týká práce s externími soubory. Pro její pochopení je třeba se předem obeznámit s odstavcem 4.4. v PP.

Každý externí soubor, jehož identifikátor je v hlavičce programu, musí mít určenou adresu svého ovládacího programu. Volba ovládacího programu určuje, zda soubor bude vstupní nebo výstupní a na jakém zařízení bude realizován.

Adresy ovládacích programů se určují:

A. Implicitně: Pro soubor INPUT čtení po řádcích z klávesnice, pro OUTPUT výstup na obrazovku, pro ostatní soubory výstup na tiskárnu.
B. Explicitně: V hlavičce programu podle PP odst. 4.4.
C. Dodatečně: Níže popsaným způsobem v příkaze RUN.

Tyto možnosti jsou uspořádány podle vzrůstajících priorit. Pro každý soubor platí, že určení adresy ovládacího programu s větší prioritou ruší určení s menší prioritou.

Dodatečné určování adres ovládacích programů v příkaze RUN se řídí těmito pravidly:

1. Za písmenem R může následovat posloupnost hexadecimálních adres ovládacích programů oddělených čárkami. Tyto adresy odpovídají po řadě „jedna k jednomu" identifikátorům externích souborů uvedeným v hlavičce programu.
2. Adresu ovládacího programu souboru lze vynechat – nesmí se však vynechat příslušná oddělovací čárka (kromě bodu 3.). Vynecháme-li dodatečné určení adresy, použije se určení s menší prioritou (explicitní nebo implicitní).
3. Chceme-li vynechat adresy ovládacích programů posledních několika souborů (jednoho nebo více, případně všech), není třeba mezi vynechanými adresami psát ani oddělovací čárky. Jinými slovy, čárky na konci příkazu lze vynechat.

Příklady:

= R – spouští běh programu. Platí explicitní nebo implicitní určení adres ovládacích programů. Není-li explicitní určení adres použito, jde standardní výstup na obrazovku a standardní vstup se čte po řádcích z klávesnice.

Nechť v hlavičce programu je uvedeno:

PROGRAM A (SPEC: 22013, INPUT, TISK, POM: 21653, OUTPUT)

Nechť běh programu se spouští příkazem:

= R, 6F00, ,6F3D

Potom:

– dodatečná adresa pro soubor SPEC není uvedena,

použije se explicitní adresa 22013 (dekadicky),
- pro soubor INPUT se použije dodatečná adresa 6F00H,
- pro soubor TISK není uvedena dodatečná ani explicitní adresa, použije se implicitní adresa pro výstup na tiskárnu,
- pro soubor POM má dodatečná adresa 6F3DH přednost před explicitní adresou 21653 (dekadicky), použije se adresa 6F3DH,
- pro soubor OUTPUT není dodatečná adresa uvedena (a ani čárka podle pravidla 3.), použije se implicitní výstup na obrazovku.

Dodatečné určení adres ovládacích programů souborů při spuštění programu platí pouze pro tento běh.

4.4. Příkaz WRITE – W

Příkaz WRITE slouží k nahrání zdrojového programu na kazetu. Umožní nám to uchovat zdrojový program po dobu vypnutí počítače na kazetě a později se k němu vrátit.

Nahráním zdrojového programu se vytvoří tzv. soubor na kazetě. Každý soubor je označen číslem z intervalu 0 až 7FH. Toto číslo nahrávaného souboru je (jediným) argumentem příkazu WRITE.

Před použitím příkazu WRITE je třeba připojit k počítači kazetový magnetofon a vyhledat na kazetě volné místo. Pak zadáme příkaz WRITE. Na obrazovce se objeví nápis „PRIPRAV KAZETU". Zapneme **nahrávání** na magnetofonu a stiskneme libovolnou klávesu počítače.

Čas nahrávání závisí na délce zdrojového programu. Po nahrání se přihlásí řídicí systém a lze zastavit

magnetofon. Nahraný program zůstane i nadále v paměti počítače.

Doporučení:

1. Editujeme-li dlouhý program, je vhodné si jej občas nahrát. Zabezpečíme se tak např. proti výpadku napájení.
2. Je vhodné si vést přesnou evidenci souborů nahraných na kazetách. U každého souboru si poznamenáme jeho číslo.

Příklad:

=W7 – nahraje zdrojový program do souboru s číslem 7.

4.5. Příkaz READ – U

Příkaz slouží k nahrání souboru se zdrojovým programem do paměti počítače. Je-li v paměti starý zdrojový program, pak se nahráním zruší.

Příkaz má jediný argument, číslo souboru, který se má nahrávat. Z kazety se nahraje jen soubor se stejným číslem.

Před použitím příkazu READ je třeba k počítači připojit magnetofon a na kazetě vyhledat místo, za nímž je nahraný soubor. Pak zadáme příkaz READ. Na obrazovce se objeví nápis „PRIPRAV KAZETU". Stiskneme libovolnou klávesu počítače a zapneme přehrávání na magnetofonu.

Během nahrávání se na obrazovce objevují čísla souborů, jejichž hlavičky jsou zrovna čteny. Čte-li se zadaný soubor, obrazovka roluje nahoru.

Objeví-li se před návratem do řídicího systému na obrazovce nápis „CHYBA CTENI Z KAZETY", je třeba nahrávání zopakovat.

Příklad:
=U 7 – čte zdrojový program ze souboru číslo 7.

4.6. Příkaz SAVE – S

Příkaz SAVE umožňuje nahrát přeložený kód na kazetu. Argumentem příkazu SAVE je číslo vytvářeného souboru.

Příkaz SAVE lze použít jen tehdy, když v paměti počítače je bezchybně přeložený kód získaný překladem (COMPILE). Nelze nahrát na kazetu kód získaný provedením příkazu LOAD.

Způsob práce s kazetovým magnetofonem je stejný jako u příkazu WRITE.

Příklad:
= S6C – nahraje kód do souboru s číslem 6CH

4.7. Příkaz LOAD – L

Příkaz LOAD nahrává soubor obsahující přeložený kód z kazety do počítače. Argumentem příkazu je číslo nahrávaného souboru. Nahrát lze pouze soubory vytvořené příkazem SAVE.

Pokud byl v paměti před nahráváním nějaký kód, zruší se.

Způsob práce s magnetofonem je stejný jako u příkazu READ.

Příklad:
= L6C – nahraje soubor s kódem číslo 6CH.

4.8. Příkaz BASIC – B

Zadáním příkazu BASIC se systém vrátí do Basicu. Příkaz BASIC nemá žádné argumenty. Jeho provedení

zruší případný zdrojový program nebo kód v paměti. Pokud byl v Basicu před přechodem do systému Pascal vytvořen nějaký program, je také zrušený.

Příklad:
= B

4.9. Příkaz INITIALIZE – I

Zadáním příkazu INITIALIZE vyvoláme inicializaci celého systému Pascal. Postup inicializace je stejný jako po zadání příkazu ROM v Basicu.

Při inicializaci je třeba znovu nahrát z kazety soubor s překladačem. Pokud v paměti byl zdrojový program nebo kód, zruší se.

Inicializace je nutná, došlo-li k přepsání překladače nebo systémových proměnných v paměti.

Příklad:
= I

4.10. Příkaz TOP – T

Příkazem TOP lze shora omezit oblast paměti využívanou Systémem Pascal. Standardně se používá paměť do adresy 79FFH.

Argumentem příkazu TOP je adresa, od níž nahoru nebude editor, překladač ani běžící program v Pascalu nic zapisovat. Tato adresa musí být z intervalu 500H až 7A00H.

Použitím příkazu TOP získáme oblast, do níž lze ukládat vlastní strojové podprogramy a data, zasahovat standardní procedurou POKE nebo příkazem řídícího

systému MEMORY. Takto zadané vymezení platí až do nového použití příkazu TOP nebo do inicializace systému.

Zadáme-li příliš malou velikost pracovní oblasti Systému Pascal, může dojít:

- při editaci k jejímu ukončení, pokud zdrojový program se nevejde do pracovní oblasti;
- při překladu k některé fatální chybě, pokud překladač má nedostatek paměti;
- při překladu k chybě 170, není-li dostatek adresového prostoru pro přeložený program:
- při běhu k chybě 16 nebo 50, při vyčerpání paměti.

Příklad:

= T7000 – paměť přepisovaná Systémem Pascal končí na adrese 6FFFH.

= T7A00 – Systém Pascal využívá celou paměť, iniciální stav.

4.11. Příkaz MEMORY – M

Příkaz slouží k prohlížení a modifikování obsahu operační paměti. Argument příkazu udává, od jaké adresy se má pracovat.

Po zadání příkazu se vypíše obsah byte paměti na zadané adrese. Pak:

- napíšeme-li 2 hexadecimální číslice, zapíšou se na poslední vypsanou adresu a zobrazí se obsah buňky s adresou o 1 větší;
- stiskneme-li mezeru, obsah zobrazené buňky paměti se nezmění a zobrazí se obsah buňky s adresou o 1 větší;
- stiskneme-li klávesu tabulátoru, obsah zobrazené buňky paměti sa nezmění a zobrazí se obsah buňky s adresou o 1 menší;

– stiskneme-li klávesu CR, obsah zobrazené buňky se nezmění a činnost příkazu se ukončí.

Příklad:
= M101F
101F 2D – 47
1020 A3 – 07
1021 15 –
=

Uživatel napsal M101F CR 47 07 CR. Obsah adresy 101FH se změnil z 2DH na 47H, obsah adresy 1020H z A3H na 7H, na adrese 1021H zůstalo 15H.

Pozor: Důrazně varujeme před náhodným přepisováním paměti. Může dojít ke zhroucení systému. Systém se bezpečně a důkladně zhroutí také při pokusu o přepis paměti ROM.

4.12. Příkaz GO – G

Příkaz GO slouží ke spuštění programu ve strojovém kódu. Argumentem příkazu je adresa, na níž začíná program. Program musí být ukončen instrukcí RET.

Příklad:
= G6F30 – spustí program začínající na adrese 6F30H.

Pozor: Chyba ve spouštěném programu může způsobit zhroucení systému. Příkaz GO se nehodí ke spouštění přeložených programů v Pascalu.

4.13. Příkaz ZVUK – Z

Zadáním příkazu ZVUK vypnete zvukový signál, který doprovází stisk každé klávesy. Opětovným zadáním

stejného příkazu se zvukový signál zapne. Příkaz nemá argumenty.
Příklad:
= Z

4.14. Příkaz NAHRÁVÁNÍ – N

Příkaz slouží k umožnění načtení souboru pořízeného na magnetofonu s opačnou polaritou. V takovém případě zadáme příkaz NAHRÁVÁNÍ s argumentem. Argument 0 uvede systém do původního stavu.

Příkaz NAHRÁVÁNÍ doporučujeme použít při problémech s nahráváním souboru pořízeného na magnetofonu jiného typu.
Příklad:
= N1 – před čtením souboru nahraného s opačnou polaritou
= N0 – před čtením bez obrácení polarity.

5. Rozdělení paměti

Adresový prostor 64KB procesoru počítače PP-01 je při práci v systému Pascal pokryt takto:

0 až 40KB Paměť RAM
40KB až 48KB Využito pro zobrazování (VIDEO-RAM)
48KB až 64KB Přídavný paměťový modul „PP-01 Pascal"

Využití paměti RAM je patrné z následující tabulky:

0000H – 003FH Vyhraženo pro ošetření přerušení a instrukcí RST i.

0040H – 025FH Proměnné systému Pascal.
0260H – xxxx Zdrojový program vytvořený editorem.
xxxx – 79FFH Pracovní oblast překladače a běžícího přeloženého programu. Zde zůstane po překladu kód.
7A00 – 9FFFH Překladač – čte se z kazety při inicializaci.

6. Restarty a přerušení

Systém Pascal používá instrukce RST 0, RST 1, RST 2 a RST 3. Proto není dovoleno přepsat paměť v úseku 0 až 1FH.

Přerušení 4 až 7, instrukce RST 4 až RST 7 a paměť v úseku 20H až 3FH je dovoleno libovolně používat.

7. Programy ve strojovém kódu

V rámci systému Pascal lze používat i programy ve strojovém kódu, vytvořené např. v systému MEDA. Tyto programy mohou obsluhovat nestandardní periferní zařízení nebo provádět zvlášť časově kritické výpočty. Způsob volání podprogramů ve strojovém kódu z Pascalu je popsán v PP odst. 4.7.3.

Prostor pro umístění podprogramů ve strojovém kódu získáme vymezením soukromého úseku paměti na konci oblasti pro překlad a běh programu. K tomu slouží příkaz TOP. Nedoporučujeme umísťovat podprogramy ani žádné jiné údaje mimo tento úsek.

Pro přenos strojových programů lze použít rozšířených příkazů LOAD a SAVE. Mají tento tvar:

Lč zadr

Sč odadr doadr

Příkaz LOAD přečte z kazety soubor číslem **č** a uloží jej do paměti počínaje adresou **zadr.**

Příkaz SAVE zapíše na kazetu do souboru s číslem **č** obsah paměti od adresy **odadr** do adresy **doadr.** Velikost ukládané oblasti se zaokrouhlí nahoru na počet bytů dělitelný 128.

Je třeba rozlišovat mezi výše uvedenými příkazy pro práci se strojovým kódem a příkazy LOAD a SAVE použitými pro nahrávání přeloženého kódu z Pascalu. Soubory vytvářené v těchto dvou případech nejsou kompatibilní.

Stejně tak nelze číst příkazem READ soubor vytvořený příkazem SAVE a příkazem LOAD soubor vytvořený příkazem WRITE.

8. Alokace generovaného kódu

Kód vytvářený překladačem se umísťuje na jisté místo do pracovní oblasti překladače v paměti. Kromě fyzického umístění v paměti je však třeba také uvažovat, do jakého adresového prostoru je kód logicky alokován tj. na jakých adresách musí být umístěn, aby mohl být spuštěn.

Za normálních okolností, tedy po použití příkazu COMPILE tak, jak byl popsán v odst. 4.2., je vytvářený kód logicky alokován přesně tam, kam je fyzicky zapisován. V důsledku toho kód může být po překladu ihned spuštěn, bez jakéhokoliv přemísťování v paměti.

Vygenerovaný kód potřebuje pro svůj běh souvislý úsek operační paměti. V tomto úseku je od spodního konce uložen kód, od horního konce se vymezuje prostor pro data. Při běhu se nad kódem vytváří „halda", pod daty je umístěn strojový zásobník. Obě tyto dynamické oblasti mohou růst. Pokud se při běhu střetnou, dojde k běhové chybě – vyčerpání paměti.

kód	halda	–)	(–	zásobník	data
ZACADR					KONADR

Na obrázku je oblast pro běh vymezena adresami ZACADR a KONADR. Tyto dvě adresy jsou parametry překladu. Standardně se ZACADR rovná adrese místa, od něhož se do paměti zapisuje kód, KONADR je konec pracovní oblasti, zadaný příkazem TOP (standardně je to 79FFH).

Zkušenému uživateli je poskytnuta možnost změnit ZACADR a KONADR. Opakujeme, že tato změna neovlivní místo, kam se kód zapisuje při překladu. Ovlivní to, na jakém místě kód může běžet. Proto změní-li se ZACADR, musí být kód mezi překladem a během přemístěn.

K přemístění kódu dojde tehdy, když kód nejprve nahrajeme na kazetu příkazem SAVE a poté jej načteme příkazem LOAD. (Použijeme formu příkazů podle 4.6. a 4.7.) Přímé přemístění kódu bez nahrávání není možné.

Změnu ZACADR a KONADR zadáme tak, že za příkaz COMPILE uvedeme další argumenty ve tvaru:

;ZACADR;KONADR

nebo jen:

;ZACADR

případně jen:

;;KONADR

kde ZACADR a KONADR jsou příslušné hexadecimální adresy. Neuvedená adresa se nezmění.

Příklad:

= C;;7200 – bez změny přepínačů, KONADR se změní na 7200H

=CS+;270 – přepínač S aktivován, změněno ZACADR

= C;1000;7700 – změněno ZACADR i KONADR

Možnosti smysluplného využití manipulace se ZACADR a KONADR:

1. Snížení KONADR

Mezi novou hodnotou KONADR a koncem paměti Systému Pascal (zadán příkazem TOP) vznikne oblast, kterou sice přepisuje překladač, ale nikoliv běžící program. Lze ji např. využít pro nestandardní manipulaci s daty v běžícím programu. Tato data budou ovšem při překladu zničena.

2. Snížení ZACADR

Lze použít pro práci s programem se zvlášť velkými paměťovými nároky při běhu. Po úspěšném překladu se ZACADR = 270H smažeme zdrojový program a kód nahrajeme na kazetu. Po nahrání kódu zpět do počítače kód bude pracovat i v oblasti, kterou původně zabíral zdrojový program.

POZOR: Program přeložený se změněným ZACADR nelze po nahrání na kazetu a načtení z kazety opětovně nahrát na kazetu.

Závěrem důležité upozornění:

Manipulace s adresami ZACADR a KONADR je závažným zásahem do systému. Systém se zde neumí dost účinně bránit proti nesmyslnému počínání operátora. Zejména následující chyby mohou vést k neadekvátní odezvě systému, včetně zhroucení:

- zadání hodnoty ZACADR menší než 270H,
- hodnoty ZACADR a KONADR mimo oblast RAM nebo velmi blízko u sebe,
- zadání hodnoty KONADR větší než 7A00H,
- spuštění editoru po přepsání zdrojového textu,
- zadání příkazu SAVE na již přemístěný kód.

9. Ovládání nestandardních periférií

Uživatel může pracovat se soubory na nestandardních periferiích, pokud si vytvořil ovládací program periférie, umístil jej do operační paměti a jeho adresu předal Pascalu při překladu (v hlavičce programu) nebo při spuštění běhu (v příkazu RUN).

Na periferním zařízení lze realizovat buď vstupní nebo výstupní soubor. Za akce spojené s otevřením souboru zodpovídá uživatel, procedury RESET nebo REWRITE jsou sice povinné, ale periférií neinicializují.

Specifikace ovládacích programů:

1. Program přenáší mezi Pascalem a periférií 1 byte, bez ohledu na velikost položky souboru.
2. Pro vstupní soubor program přečte byte do registru A.
3. Pro výstupní soubor program odešle byte z registru C.
4. Po čtení ze vstupního souboru se vrátí příznak CARRY rovný 1, právě když je EOF. Obsah registru A je při CARRY = 1 libovolný.
4. Program nesmí změnit žádné jiné registry.
5. Program je ukončen instrukcí RET.

Vhodným prostředkem pro vytváření složitějších ovládacích programů je systém MEDA. Poznámky o umístění strojových programů jsou v kap. 7.

Chceme-li připojit nestandardní tiskárnu, napíšeme podprogram pro odeslání byte v registru C na tuto tiskárnu. Podprogram uložíme na místo vymezené v paměti příkazem TOP a jeho adresu zapíšeme na adresu 41H. Na tuto tiskárnu se pak může odesílat protokol o překladu, informace o běhových chybách (DUMP) a s tiskárnou bude spolupracovat ovládací program „C" (viz PP 4.4.2.).

10. Kompatibilita

Spolupráce programů v Pascalu a Basicu (např. voláním podprogramů) není možná.

Program v Pascalu může volat podprogramy ve strojovém kódu přes standardní proceduru CALL (viz PP, odst. 4.7.3.). Předávání parametrů umožňují standardní procedura POKE a funkce PEEK.

Přenos dat mezi systémem MEDA a Pascal je možný pomoci příkazů LOAD a SAVE s udáním adres (viz kap. 7.). Takto lze snadno přenést strojové podprogramy do Pascalu.

11. Uložení zdrojového programu

Na začátku oblasti určené pro zdrojový program jsou 2 byty s hodnotami 0AH. Dále následují zdrojové řádky, každá je ukončena 0DH a 0AH. Celý program je ukončen dvěma 0AH.

Zadáme-li omylem příkaz EN místo E, nesmíme v editoru nic napsat a okamžitě jej opustíme. Pak pomocí příkazu MEMORY přepíšeme obsah adres 262H, 263H a 264H hodnotami 7CH. Přejdeme-li nyní do editoru, budou přepsané jen první 3 znaky.

12. Chybová hlášení řídicího systému

Zadáme-li chybný příkaz, pak místo jeho provedení se na obrazovce objeví některé z těchto hlášení:

NEZNAMY PRIKAZ – první znak na řádce není jménem žádného příkazu.

KOD NESPRAVNE UMISTEN – při překladu byla změněna hodnota ZACADR. Před spuštěním nutno kód nahrát na kazetu a zpět do počítače.

V PAMETI NENI KOD – běh programu nebo nahrávání

kódu lze spustit pouze pokud v paměti je kód získaný bezchybným překladem nebo nahráním z kazety.
CHYBA V SYNTAXI – chyba v argumentech příkazu.
VELKE CISLO – uvedeno číslo souboru větší než 7FH.

Přepíšete-li si při práci tu část Systému Pascal, která se nahrává z kazety, objeví se hlášení:
CHYBA V RAM CASTI
V takovém případě si příkazem WRITE nahrejte zdrojový program na kazetu a proveďte reinicializaci systému příkazem INITIALIZE.

Dojde-li k poškození externího modulu PP-01 PASCAL, hlásí se:
CHYBA V ROM CASTI
Modul je pak nutno odeslat do opravny.

13. Nahrání překladače na kazetu

Vzhledem k omezené spolehlivosti magnetických pamětových médií (zvláště magnetofonových kazet) je vhodné si pořídit kopie té části překladače, která se nahrává z kazety.

Kopii vytvoříme po inicializaci systému příkazem:
S5D 7A00 9FFF

SYSTÉM PASCAL NA PP-01
Editor

Obsah:

1. Úvod

Editor je samostatná část systému Pascal sloužící k vytváření zdrojových programů v Pascalu a k provádění změn v těchto programech.

Editor je do značné míry shodný s editorem v systému MEDA.

V editoru lze používat všechny ASCII znaky z klávesnice PP-01.

2. Práce s editorem

Editor sa vyvolává z řídicího systému příkazem E nebo EN. Po ukončení editace se vracíme zpět do řídicího systému.

Po zavolání editoru se na obrazovce vypíše hlavička. Editujeme-li již existující program, vypíše se také jeho začátek. V levém horním rohu textu se objeví kurzor.

Polohu kurzoru je třeba sledovat po celou dobu editace. Zpravidla na ní závisí funkce příkazů.

Základy editace lze nejlépe zvládnout tak, že si v praxi postupně vyzkoušíme funkci všech níže popsaných příkazů editoru.

3. Rozdělení obrazovky

V editoru je obrazovka rozdělena do 2 částí:

1. část obsahuje hlavičku, která nás neustále informuje o poloze kurzoru a rozsahu souboru. Má tvar:

STRANA:ss RADKA:rr ZNAK:zz
OD:XXXX DO: YYYY PROGRAM

Stránky, řádky na stránce a znaky na řádce se číslují od nuly.

2. část obsahuje vytvářený nebo upravovaný zdrojový program.

4. Příkazy editoru

Příkazy lze v editoru zadávat těmito způsoby:
A/ pomoci speciálních kláves (např. DEL, ADR, šipky)
B/ pomoci funkčních kláves
C/ pomoci klávesy CTRL a dalšího znaku

Funkční klávesy tvoří horní řadu na klávesnici a číslují se od 1 do 14 zleva doprava. Značíme je F1 až F14.

Některé řídicí klávesy jsou označeny grafickými symboly. Klávesa tabulátoru je označena šipkou doprava ukončenou svislou čarou a je vlevo na klávesnici.

Většinu příkazů lze zadat několika způsoby. Stiskneme-li příkazovou klávesu, na obrazovce se neobjeví znak z klávesy, nýbrž výsledek provedení příkazu. Použijeme-li k zadání příkazu způsob C/, je třeba podržet klávesu CTRL a současně stisknout další klávesu. Tento případ budeme v textu značit hvězdičkou před písmenem zadávaným současně s CTRL. Označení kláves provádějících příkazy je v nadpisech odstavců. V závorkách je alternativní způsob zadání.

Provedení příkazu editoru je okamžité. Podržíme-li klávesu, akce vyvolávaná klávesou se začne rychle opakovat (tzv. autorepeat).

Příkazy lze rozdělit do pěti skupin:
1. Vkládání textu
2. Přesuny kurzoru
3. Rušení znaků
4. Stránky
5. Ostatní příkazy

4.1. Vkládání textu

Píšeme-li na klávesnici text, objevuje se na obrazovce v místě, kde je kurzor. Kurzor se přitom posouvá

doprava. Pokud bychom kurzor nastavili na již napsaný text, pak zapisované znaky by tento text **přepsaly**. Toho lze využít při opravování textu.

4.1.1. Zrušení napsaného znaku – DEL (*O)

Stisknutím této klávesy rušíme znak vlevo od kurzoru. Píšeme-li text, zrušíme tak posledně napsaný znak na řádce.

4.1.2. Ukončení řádky – CR (*M)

Pokud jsme dopsali nový řádek textu a chceme přejít na další, stiskneme klávesu CR. Kurzor se přesune na začátek následující řádky.

Klávesou CR musíme ukončit i poslední řádku zdrojového programu, jinak by ji kompilátor nepřijal.

4.1.3. Tabulátor – klávesa tabulátoru (*I)

Tabulátor vloží do textu na místo, kde je kurzor, 1 až 4 mezer. Kurzor se přesune za poslední mezeru. Počet mezer je takový, aby nová pozice kurzoru na řádce byla dělitelná 4 (viz údaj ZNAK v hlavičce).

Tabulátor využijeme všude tam, kde text má být formátován po sloupcích. Zrychluje odsazování programu od levého okraje.

4.1.4. Vkládání znaků dovnitř řádky – F4 (*A)

Dosud popsané metody dovolovaly vytvářet nové řádky a přepisovat znaky na řádce. Chceme-li dovnitř (nebo před začátek) existující řádky vložit nějaké znaky,

stiskneme klávesu F4 (*A). Píšeme-li pak text, zbytek řádky se nepřepisuje, ale posouvá doprava. Vkládání ukončíme opětovným stisknutím F4 (*A).

Při vkládání znaků dovnitř řádky lze chybně napsaný znak zrušit pomoci DEL. Zadáme-li jakýkoliv jiný řídicí příkaz, neprovede se, ale režim vkládání se ukončí.

4.2. Přesuny kurzoru

Text lze opravovat nebo psát pouze v místě, kde je kurzor. Proto je občas třeba kurzor přemístit.

Zapamatujte si toto pravidlo: kurzor lze vždy přemístit jen na napsaný text nebo bezprostředně za konec řádky.

4.2.1. Kurzor doprava – šipka doprava (*X)

Kurzor se posune o 1 znak doprava. Je-li kurzor za posledním znakem řádky, nestane se nic.

4.2.2. Kurzor doleva – šipka doleva (*H)

Kurzor se posune o 1 znak doleva. Je-li kurzor na prvním znaku řádky, nestane se nic.

4.2.3. Kurzor dolů – šipka dolů (*K)

Kurzor se přesune na první znak následující řádky. Pokud následující řádka není na obrazovce, objeví se. Pokud kurzor byl na poslední řádce, nestane se nic.

Máme-li kurzor na poslední řádce textu a chceme-li vytvořit další řádku, nestiskneme šipku dolů, ale klávesu CR.

4.2.4. Kurzor nahoru – šipka nahoru (*Z)

Kurzor se přemístí na první znak předchozí řádky. Pokud předchozí řádka nebyla na obrazovce, objeví se. Pokud kurzor byl na první řádce textu, nestane se nic.

4.2.5. Kurzor na začátek řádky – LF (*J)

Kurzor se přesune na začátek té řádky, na níž zrovna je.

4.3. Rušení znaků

Příkazy pro rušení znaků pracují vždy v rámci jedné řádky.

V odst. 4.1.1. jsme se už seznámili s příkazem DEL, který ruší znak před kurzorem (tj. vlevo od kurzoru).

4.3.1. Zrušení znaku, na němž je kurzor – ADR

Klávesou ADR zrušíme znak, na němž je kurzor. Pokud kurzor byl za posledním znakem řádky, nestane se nic.

Příkaz ADR používáme obvykle při opravování textu, zatímco příkaz DEL při vkládání textu.

4.3.2. Zrušení znaků do konce řádky – F6 (*B)

Příkaz zruší znaky od kurzoru do konce řádky.

4.3.3. Zrušení řádky nebo její části – F7 (*C)

Příkaz zruší znaky od kurzoru do konce řádky a za zbylé znaky připojí následující řádku. Speciálně, pokud kurzor byl na prvním znaku řádky, pak řádka zmizí celá.

Tento příkaz umožňuje vytvořit řádku delší než obrazovka. Z takovéto řádky se však zobrazuje jen prvních 31 znaků.

Chceme-li zrušit několik řádek textu, pak přesuneme kurzor na začátek první z nich (příkaz LF) a opakovaně rušíme řádky pomoci F7 (*C).

4.4. Stránky

Uživatel editoru si pro své pohodlí může editovaný text rozdělit na stránky. Údaj STRANA v hlavičce nás

neustále informuje, na které stránce je zrovna kurzor. První řádek na stránce začíná oddělovačem stránek, který se zobrazí jako „vlnka". Zrušíme-li tento znak, stránka se stane pokračováním předchozí stránky.

4.4.1. Nastavení nové stránky – F2 (*S)

Stisknutím klávesy F2 (*S) sdělíme editoru, že řádek, na němž je kurzor, je začátkem nové stránky. Číslo stránky v hlavičce se stisknutím F2 (*S) zvětší o 1.

4.4.2. Přechod na novou stránku – F3 (*Y)

Kurzor se přesune na začátek následující stránky (tj. stránky následující za tou, na níž je kurzor). Obsah stránky se vypíše na obrazovce.

4.4.3. Přechod na předchozí stránku – F1 (*W)

Stisknutím této klávesy se kurzor přesune na začátek stránky. V případě, že už byl na začátku stránky, přesune se na začátek stránky **předchozí.** Stránka s kurzorem se zobrazí.

4.4.4. Přechod na nultou stránku – F11 (*D)

Na obrazovce se vypíše nultá stránka textu, kurzor bude na jejím začátku.

Doporučujeme editovaný zdrojový program členit na stránky podle jeho vnitřní logické struktury. Stránka by neměla být o mnoho větší než obrazovka. Příkazy F1 a F3 nám dovolí rychlé prohlížení textu.

4.5. Ostatní příkazy editoru

4.5.1. Vložení prázdné řádky – F5 (*N)

Chceme-li do textu vložit novou řádku, najedeme kurzorem na tu řádku textu, před niž má nová řádka vzniknout. Pak stiskneme F5 (*N).

4.5.2. Výměna řádek – F8 (*V)

Příkaz přehodí řádku, na níž je kurzor, s předchozí řádkou. Kurzor se přesune spolu s řádkou o 1 pozici výše.

4.5.3. Vyhledání řetězce – F13 (*U)

Po zadání příkazu F13 (*U) se editor zeptá, jaký řetězec znaků chceme vyhledat. Napíšeme jej a ukončíme klávesou CR. Editor jej začne hledat v textu od místa, kde je kurzor. Najde-li jej na nějaké stránce, pak tuto stránku zobrazí na obrazovce. Pokud řetězec není v textu nalezen, zobrazí se stránka, na níž byl kurzor.
Příkaz:
Na dotaz VYHLEDAT: zadáme řetězec „ANO". Editor jej najde ve slově „JANO", ale nikoliv v dvojici řádek:

SPISSKA
NOVA VES

protože zde je řetězec rozdělen oddělovačem řádek.

Příkaz vyhledání řetězce dovolí nejrychlejším způsobem přesunout kurzor k hledanému místu v rozsáhlém programu.

4.5.4. Zapnutí / vypnutí zvukového signálu – F14

Příkaz vypne zvukový signál, který doprovází stisk každé klávesy. Je-li signál vypnut, stisknutí F14 ho zapne.

4.5.5. Nastavení barvy – F12 (*F)

Po zadání příkazu se editor zeptá na barvu podkladu a písma. Barva se zadá čísly v rozsahu 0 až 7.

4.5.6. Ukončení editace – F9 (*E)

Příkaz ukončí editaci programu. Obrazovka se smaže a na první řádce se objeví nápis:

OD: xxxx DO: yyyy

Uživatel se takto dozví, kde je v paměti uložen zdrojový program.

Vzápětí se svým nápovědným znakem „=" přihlásí řídicí systém.

5. Přerušení editace

Stiskneme-li klávesu přerušení INT 0, ihned se ukončí práce editoru a přihlásí se řídicí systém. Není zaručeno, že zdrojový program byl zanechán v podobě přijatelné pro překladač.

6. Závěrečná doporučení

Editujeme-li program v Pascalu, není vhodné příliš šetřit na komentářích a mezerách při indentaci (odsazování) příkazů. To neplatí pouze pro programy kritické velikosti.

Při editování dlouhého programu bývá občas vhodné vytvořený program nahrát na kazetu. Zabezpečíme se tak např. proti výpadku sítě.

Vyrobte si proužek papíru vysoký asi 2 cm a rozdělený na 14 políček. Tento proužek položíte nad funkční klávesy počítače. Na jednotlivá políčka si napište tato hesla:

předchozí stránka
nastavení stránky
následující stránka
vkládání znaků
prázdná řádka
smaž do konce řádky
smaž řádku
prohození řádek
konec

---

začátek textu
barva
vyhledání
zvuk

Příloha A: Tabulka příkazů řídicího systému:

E	Editor, „N" – smaže starý program
C	Compile, start překladu, X+,X– nastavuje přepínače
R	Run, start běhu programu
W	Write, nahraje program na kazetu
U	Read, čte program z kazety
S	Save, nahraje kód na kazetu
L	Load, čte kód z kazety
B	Basic, přechod do Basicu
T	Top, nastavení konce paměti pro překladač
I	Initialize, reinicializace a nové nahrání systému
M	Memory, prohlížení a přepisování paměti
G	Go, start podprogramu ve strojovém kódu
Z	Zvuk, vypnutí/zapnutí zvukového signálu
N	Nahrávání, 1 – obrácená polarita, 0 – normální polarita.

Příloha B: Tabulka příkazů editoru:

šipka doprava	*X	kurzor vpravo
šipka doleva	*H	kurzor vlevo
šipka nahoru	*Z	kurzor na předchozí řádku
šipka dolů	*K	kurzor na další řádku
LF	*J	kurzor na začátek řádky
tabulátor	*I	tabulátor
DEL	*O	zruš znak před kurzorem
ADR		zruš znak pod kurzorem
F6	*B	zruš do konce řádky
F7	*C	zruš řádku
F4	*A	vkládání znaků
CR	*M	konec řádky
F5	*N	vlož řádku
F8	*V	prohození řádek
F2	*S	nastavení začátku stránky
F1	*W	předchozí stránka
F3	*Y	následující stránka
F11	*D	začátek textu
F13	*U	hledání řetězce
F14		vypnutí – zapnutí zvuku
F12	*F	nastavení barvy
F9	*E	konec

Příloha C: **Odlišné varianty Systému Pascal**

Systém Pascal existuje ve dvou dalších variantách:

- ROM verze: celý systém je v pamětech EPROM místo interpretu BASICu. Není potřeba připojovat externí modul ani nahrávat druhou část z kazety. Systém se přihlásí po zapnutí počítače.
- RAM verze: celý systém se nahrává do operační paměti. Není třeba připojovat externí modul.

Hlavní odlišnost těchto variant od základní verze spočívá ve velikosti operační paměti, která je k dispozici Systému Pascal. Zatímco v základní verzi končí tato paměť adresou 79FFH, v ROM verzi je to 9FFFH a v RAM verzi 39FFH. Tím je také určena nejvyšší hodnota, kterou lze uvést jako argument příkazu TOP.

V ROM verzi nelze použít příkaz BASIC, v ROM ani RAM verzi neexistuje příkaz INITIALIZE.

ROM MODUL PASCAL
pre personálny počítač SMEP PP O1
(príručka používateľa)

AUTOR: RNDr. JANUŠ DROZD
Náklad: 5000

Rukopis neprešiel jazykovou ani redakčnou úpravou!

Vytlačili Tlačiarne SNP, Martin

ROM modul PP 01

SPC PASCAL

OPVS DOLNÝ KUBÍN

PASCAL	CJK 403 644 000 030	PN 121-87
MEDA	ČJK 403 644 000 031	PN 121-87