
OPERAČNÍ SYSTÉM
AMOS

Obsah VII. kapitoly PASCAL pod OS AMOS

VII. Pascal pod OS AMOS

Překladač jazyka PASCAL implementovaný na počítač, IQ 151 pod OS AMOS je výsledkem snahy zpřístupnit vyšší programovací jazyk na tomto počítači bez dalších nároků na periferní zařízení (pružné disky).

Překladač až na několik výjimek, které si vynutila specifika mikropočítače a nedostatek paměti RAM, vyhovuje mezinárodní normě jazyka PASCAL na úrovni 0. To znamená, že nemá implementována konformní schemata polí. Celkově se dá říci, že míra, v jaké překladač vyhovuje normě, je srovnatelná s překladači PASCALU na velkých počítačích. Žádná z výjimek z normy neomezuje podstatným způsobem programátora.

Na druhé straně obsahuje Pascal na IQ 151 řadu rozšíření, která rozšiřují oproti standardu práci se soubory, umožňují spolupráci s podprogramy ve strojovém kódu, ovládání minigrafu a některé další funkce. Repertoár standardních funkcí a procedur půjde rozšiřovat i dodatečně formou modulů nahrávaných do paměti RAM.

Překladač je jednoprůchodový, využívá tzv. semistatického způsobu alokace proměnných. Každé alokované proměnné je pevně přidělen úsek paměti. To je velmi výhodné pro velikost přeloženého programu i pro jeho rychlost. Při každé další aktivaci rekurzivní procedury se pak musí staré lokální hodnoty odsunout na zásobník. To může vést k některým odchylkám chování programu od chování podle normy (viz odst.9).

Vzhledem k rozsahu nemůže být tento manuál učebnicí PASCALU. Čtenáře, kteří PASCAL neznají, odkazujeme na knihu J.Jinoch, K.Muller, J.Vogel : Programování v jazyku Pascal, která vyšla v nakladatelství SNTL v roce 1985.

V příloze tohoto manuálu jsou syntaktické diagramy, definující standardní jazyk, jinak tento manuál uvádí kromě výjimek z normy, implementačních omezení a rozšíření jazyka také základní informace o přidělování paměti, paměťových nárocích jednotlivých typů a další údaje, užitečné při používání překladače pod OS AMOS.

Pro úspěšnou práci s Pascalem i pro porozumění této kapitole je nutná znalost prvních tří kapitol manuálu OS AMOS. Pokud uživatel potřebuje psát podprogramy v assembleru, musí se seznámit s manuálem celým.

1. Kvantitativní omezení daná implementací

- V definici výčtového typu lze uvést nejvýše 256 konstant.

- Lze použít nejvýše 6 úrovní statického vnoření procedur. (Tj. hlavní program a 5 vnořených procedur). Dynamické vnořovaní je omezeno jen velikostí paměti.

- Součet počtu procedur, počtu funkcí a počtu návěští, na něž vedou skoky ven z bloku, nemůže v celém programu přesáhnout 255.

2. Symboly jazyka

2.1. Alternativní reprezentace znaků

V souladu s doporučením normy Pascalu je možno používat alternativní zápis pro komentářové a hranaté závorky

Standardně	alternativně
{	(*
}	*)
[	(.
]	.)

Znak ↑ má dvě případné reprezentace "@" (zavináč - 40H) a "^" (stříška - 5EH).

2.2. Identifikátory a klíčová slova

V zápisu identifikátoru, resp. klíčového slova je signifikantních prvních šest znaků tj. překladač nerozlišuje mezi identifikátory resp. klíčovými slovy, které se shodují na prvních šesti znacích.

V souhlase s normou se v identifikátorech a klíčových slovech nerozlišují malá a velká písmena. V identifikátorech se smí vyskytovat také znak dolar, doporučujeme však uživatelům, aby identifikátory obsahující tento znak ve svých programech nepoužívali, některé z nich budou využívány při rozšířeních nahrávaných do paměti RAM.

Pro zlepšení čitelnosti lze dovnitř identifikátorů vkládat znak "_" (podtrhovátko), překladač tento znak ignoruje. Podtrhovátko nelze však vkládat do klíčových slov.

Např.:
Je_Tam a JETAM jsou přípustné zápisy téhož identifikátoru
GO_TO   je nepřípustný symbol
PROGRAMATOR   je klíčové slovo PROGRAM
NAPETI1 a NAPETI2 jsou přípustné zápisy téhož identifikátoru

3. Datové typy

3.1. Jednoduché datové typy

3.1.1. Typ integer

Hodnota maxint je 32767, hodnoty typu integer musí tedy ležet v intervalu -32767 až 32767. Hodnoty jsou ukládány ve dvou bytech v doplňkovém kódu.

Konstanty typu integer mohou být v pascalských programech zapsány i hexadecimálně. Hexadecimální zápis musí začínat číslicí (0-9) a musí být ukončen písmenem H. Pomocí hexadecimálního zápisu specifikujeme přímo vnitřní tvar čísla typu integer, pokud jsou zadány více než čtyři hexadecimální cifry, mají význam pouze poslední čtyři.

Při programování je nutné dbát na to, aby výsledky (i mezi výsledky) operací, prováděných nad typem integer nepřesáhly hodnotu maxint. Dojde-li k tomu, je signalizována běhová chyba 61.

Např.

1BH	Je konstanta typu integer (=27)
32000 + 1000 - 1000	při vyčíslování tohoto výrazu dojde k běhové chybě
A5FH	Není hexadecimální zápis čísla (nezačíná číslicí)

3.1.2. Typ real

Část konstanty typu real uvedená před desetinnou tečkou musí být v absolutní hodnotě menší než 32768 (tj. musí být číslem typu integer).

Lze tedy psát : 12345.678E13
ale nikoliv : 123456.78E12

Hodnoty typu real jsou ukládány ve čtyřech bytech. Nejmenší kladné zobrazitelné číslo je přibližné 1.18E-38, největší je přibližně 3.4E+37. Mantisa čísla zabírá 23 bitů, což poskytuje přesnost zobrazení zhruba 6 desetinných míst.

3.1.3. Typ char

Hodnotou typu char může být všech 256 znaků zobrazitelných na IQ 151. Hodnota typu char se ukládá v jednom bytu. Uspořádání znakové množiny a ordinální čísla znaků jsou určena kódem ASCII.

3.1.4. Typ boolean

Hodnoty typu boolean se ukládají v jednom bytu. Ordinální hodnota false je 0, true je 1.

3.1.5. Výčtové typy

Hodnoty výčtových typů se ukládají v jednom bytu. Výčtový typ tedy smí mít mohutnost nejvýše 256 hodnot.

3.1.6. Typy interval

Reprezentace hodnoty typu interval je shodná s reprezentací hodnoty jeho hostitelského typu.

3.2. Strukturované datové typy

3.2.1. Pakování typů

Definice strukturovaného typu může být prefixována vyhrazeným slovem packed (prefix vyjadřuje přání, aby hodnoty typu byly uloženy úsporně, "pakovaně"). Uvedení prefixu nemá však žádný vliv na zobrazení hodnot typů ani na kompatibilitu typů. Lze tedy dosazovat pakované řetězce do nepakovaných a naopak, složka pakované struktury smí být skutečným parametrem procedury, který je volán referencí, atd. .

Protože k pakování hodnot strukturovaných proměnných nedochází, jsou standardní procedury pack a unpack zbytečné a nejsou tedy implementovány.

3.2.2. Typy pole

Prvky pole jsou při uložení do paměti uspořádány lexikograficky podle indexů. Velikost paměťového nároku proměnné typu pole je roven velikosti paměťového nároku složky pole vynásobenému počtem složek.

3.2.3. Typy záznam

Paměťové nároky typu záznam se rovnají součtu velikostí nároku pevné části a nároku té větve variantní části, která má největší nárok.

Položky záznamu se ukládají za sebou v pořadí jejich deklarace. Výjimku tvoří případ, kdy uvádíme několik jmen položek stejného typu oddělených čárkami. Tyto položky se pak uloží v obráceném pořadí.

Např.
Položky typu záznam record A,B: real; C: integer end
se ukládají v pořadí B,A,C. Jedna hodnota tohoto typu zabírá
10 bytů.

3.2.4. Typy množina

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

Hodnota typu množina se zobrazuje ve 32 bytech paměti. Jednotlivé bity reprezentují prvky bázového typu, prvku s ordinálním číslem nula odpovídá nejnižší bit v bytu s nejnižší adresou.

Např.
```
   set of 0..25
   set of boolean
   set of (TREFY,KARA,SRDCE,PIKY)
   set of char
   set of 1..100
```

ale nelze
```
   set of 1..1000
   set of integer
```

3.2.4. Typy soubor

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

Uložení proměnné typu soubor vyžaduje 5 bytů + velikost paměti potřebné pro uložení jedné položky souboru.

Další informace o práci se soubory jsou v paragrafu 8.

3.3. Typy ukazatel

Doménový typ typu ukazatel nesmí být typem soubor (ani nesmí takový typ obsahovat). Hodnoty ukazatelů se ukládají jako adresa ve 2 bytech.

4. Přidělování paměti proměnným a parametrům

Paměť se deklarovaným proměnným přiděluje zásadně od vyšších adres k nižším. Přidělování probíhá v tom pořadí, v jakém byly proměnné deklarovány, s výjimkou proměnných jednoho typu, jejichž deklarace jsou odděleny čárkami, těm se paměť přiděluje v opačném pořadí.

Např.
Při deklaracích

```
      var I   : integer;
          A,B,
          C   : integer;
          J,K : integer;
```

se paměť přiděluje proměnným v pořadí I, C, B, A, K a J.

Množství přidělené paměti je určeno typem proměnné.

Parametrům se paměť přiděluje od nižších adres k vyšším v tom pořadí, v jakém jsou uvedeny v hlavičce podprogramu.

Parametrům, volaným hodnotou, se přiděluje tolik paměti jako proměnné příslušného typu.
Parametrům volaným referencí se přidělují dva byty, procedurálním a funkcionálním parametrům jeden byte.

5. Výrazy

Jsou realizovány podle normy, vzhledem k tomu, že hodnota maxint je menší než je obvyklé u velkých počítačů, je nutné dávat při vyčíslování výrazů typu integer pozor na to, aby hodnota žádného podvýrazu nepřevýšila v absolutní hodnotě maxint.

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. Výsledek je typu integer.

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

6. Příkazy

6.1 Rozšíření příkazu CASE

V příkazu CASE lze specifikovat příkaz, který se má provést, pokud se hodnota selektoru větvení nerovná žádné z uvedených konstant. Před tímto příkazem uvedeme v seznamu konstant klíčové slovo ELSE. Klíčové slovo ELSE smí být v jednom příkazu CASE uvedeno nejvýše jednou.

Např.

```
      case HODNOTA of
         0     : TYP:=NULOVA;
        1,2,3  : TYP:= MALA;
        1000   : TYP:= VELKA;
        else   : TYP:= JINA
      end
```

6.2. Příkaz FOR cyklu

Při změně hodnoty řídící proměnné cyklu FOR uvnitř těla cyklu dojde k běhové chybě. V případě vypnutí běhových kontrol stejně taková změna neovlivní počet průchodů cyklem. Řídící proměnná cyklu musí být lokální v nejmenším bloku, který cyklus obsahuje.

Po normálním ukončení FOR cyklu není hodnota jeho řídící proměnné definována.

6.3. Příkaz GOTO

Skok dovnitř strukturovaného příkazu není ohlášen jako chyba, ale chování programu bude nepředvídané.

7. Procedury a funkce

7.1. Procedurální a funkcionální parametry

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

7.2. Standardní procedury pro dynamickou alokaci

Standardní procedury NEW a DISPOSE smějí mít právě jeden parametr - není tedy možné šetření paměti vyznačením hodnoty rozlišovací položky u typu variantní záznam.

Procedura DISPOSE je implementována v plné síle, včetně scelování paměti.

Mimo standardní procedury NEW a DISPOSE popsané v normě jsou k dispozici standardní procedury MARK a RELEASE. Mají jediný argument libovolného typu ukazatel.

Zavoláním procedury MARK(P) se proměnné P přiřadí adresa aktuálního vrcholu haldy. Když v průběhu dalšího výpočtu halda vzroste, pak 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 běhové chyby.

7.3. Strojové orientované standardní procedury

Pro umožnění pohodlné spolupráce pascalských programů s programem a assembleru resp. strojovém kódu slouží v Pascalu pod OS AMOS nové standardní procedury a funkce.

Při jejich popisu předpokládáme tyto deklarace:

```
type  BYTE    = 0..255;
      ADDRESS = integer;
```

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

```
function PEEK (A:ADDRESS):BYTE;
```
- vrací obsah bytu s adresou A

```
procedure POKE (A:ADDRESS; B:BYTE);
```
- zapíše do paměti na adresu A byte B

```
function DPEEK (A:ADDRESS): integer;
```
- hodnotou je hodnota typu integer, která vznikne interpretací obsahu dvou bytů s adresami A, A+1 jako čísla typu integer.

```
procedure DPOKE (A:ADDRESS;I:integer);
```
- zapíše hodnotu parametru I do dvou bytů, s adresami A a A+1

```
function INP (PORT:BYTE) : BYTE;
```
- hodnotou je 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:ADDRESS);
```
- provede podprogram ve strojovém kódu, který začíná na adrese START. Podprogram musí být ukončen instrukcí RET.

```
function FCALL (START:ADDRESS) : integer ;
```
- provede podprogram ve strojovém kódu, který začíná na adrese START. Podprogram musí být ukončen instrukcí RET, funkce vrací hodnotu typu integer, která byla v okamžiku provádění instrukce RET v dvojregistru DE.

Obě tyto standardní procedury mohou mít libovolný počet dalších parametrů typu integer :

```
procedure CALL (START:ADDRESS; P1,...,Pn:integer);

function FCALL (START:ADDRESS; P1,...,Pn:integer) : integer;
```

Pomocí nich lze spouštěnému podprogramu předat hodnoty skutečných parametrů. Hodnota posledního parametru je předávána v dvojregistru DE, hodnota předposledního v dvojregistru BC a hodnoty předchozích na zásobníku pod návratovou adresou v pořadí odzadu dopředu (Pn-2,...,P1). Jeli parametr jediný, je jeho hodnota jak v dvojregistru BC, tak i v DE.

Dále je realizována standardní funkce REF, mající jeden parametr. Jejím skutečným parametrem může být proměnná jakéhokoli v typu, funkce vrací adresu této proměnné. Mohli bychom tedy symbolicky zapsat deklaraci této funkce :

```
function REF (var PROM: <lib. typ> ): ADDRESS;
```

Tuto funkci můžeme využít např. k tomu, abychom assemblerskému podprogramu předali jako parametr adresu pascalské proměnné.

7.4. Čtení parametru z volací věty

```
procedure GETPARM (var S: <řetězec znaků> ; var L: integer);
```

Procedura kopíruje do parametru S řetězec <params> z volací věty programu (tj. část volací věty mezi prvním středníkem a "znakem" CR). V parametru L procedura vrací počet zkopírovaných znaků.

Skutečným parametrem odpovídajícím formálnímu parametru S může být libovolná proměnná typu znakový řetězec. Nechť je to např. proměnná R: array [1..N] of char , pak

Je-li N > délka řetězce <params>,
doplní se hodnota proměnné R zprava mezerami a
je-li N < délka řetězce <params>,
zkopíruje se pouze N znaků řetězce <params>.

Zavoláme-li proceduru GETPARM znovu, kopíruje se řetězec <params> opět od počátku.

7.5. Přesun kurzoru na obrazovce

```
procedure KURZOR( I,J : Integer) ;
```

Procedura nastaví pozici kurzoru na obrazovce na I-tý řádek a J-tý sloupec. Další výpis na obrazovku (tj. do souboru, jemuž je přiřazeno zařízení :CO:) bude probíhat od této pozice. Provedení procedury bezprostředně změní polohu kurzoru, proto příkaz nelze užít pro spoolovaný výstup na obrazovku.

7.6. Čtení znaku z klávesnice bez čekání

Čtení z klávesnice (resp. ze souboru, který vstupuje z klávesnice) se děje přes vstupní buffer, jehož obsah je nutno odeslat stiskem klávesy CR. Výhodou je především to, že jeho obsah můžeme až do tohoto odeslání editovat. V některých aplikacích (např. hry) je to však na závadu.

Aby bylo možné i z Pascalských programů číst z klávesnice bez čekání je k dispozici funkce KEY :

```
function KEY (var C:char) : boolean ;
```

Pokud je v okamžiku zavolání procedury stisknuto korektním způsobem nějaké tlačítko, vrátí funkce hodnotu true a v parametru C příslušný znak, pokud není stisknuto, vrátí hodnotu false.

7.7. Zastavení výpočtu programu

K zastavení výpočtu programu s výpisem zpráv o stavu výpočtu (post-mortem-dump) slouží procedura

```
procedure HALT;
```

7.8. Funkce změny typu

V některých aplikacích (zvláště ve spolupráci se strojovými podprogramy) může být na závadu přísná typová kontrola Pascalu, která je jinak účinnou ochranou programátora před mnohými chybami. Proto některé překladače obsahují jako rozšíření prostředky pro převod typů. Tyto prostředky má i Pascal pod OS AMOS.

V naší implementaci je s každým identifikátorem typu alfa jakoby spojena funkce téhož jména. Jejím parametrem může být výraz libovolného typu a tato funkce vrátí hodnotu typu alfa, která vznikne interpretací reprezentace hodnoty výrazu a paměti jako reprezentace hodnoty typu alfa.

Funkce změn typu lze užívat pouze uvnitř těchto skupin typů:
1. char, boolean, výčtový typ
2. integer, ukazatel
3. strukturované typy

Nedodržení tohoto omezení může mít pro běh programu fatální důsledky.

7.9. Ovládání minigrafu

Pascal pod OS AMOS obsahuje standardní funkce pro ovládání Minigrafu 0507 z k.p. Aritma. Jde o volání podprogramů obsažených v pamětech EPROM připojovacího modulu. Jejich funkce je podrobně popsána v manuálu k tomuto modulu.

7.9.1. Inicializace polohy pisátka

```
procedure QORG ( X, Y : integer ) ;
      procedura inicializuje polohu pisátka,
      přípustné hodnoty: X z <0,1500>, Y z <0,2100>
      o zadání přípustných hodnot parametrů se musí programátor
      postarat sám, překladač tyto meze nekontroluje
```

7.9.2. Absolutní posuny pisátka

```
procedure QMOVA ( X, Y : integer ) ;
      přesun (bez kresby) pisátka do bodu (X,Y), pisátko zůstane
      zvednuto

procedure QVECTA ( X, Y : integer ) ;
      nakreslí úsečku z dosavadní polohy pisátka do bodu (X,Y),
      pisátko zůstane spuštěno

procedure QPNTA ( X, Y : integer ) ;
      přesun (bez kresby) pisátka do bodu (X,Y) a udělání tečky v
      tomto bodě, pisátko zůstane spuštěno
```

7.9.3. Relativní posuny pisátka

```
procedure QMOVR ( XR, YR : integer ) ;
      přesun (bez kresby) pisátka z dosavadní polohy (X0,Y0) do bodu
      (X0+XR,Y0+YR), pisátko zůstane zvednuto

procedure QVECTR ( XR, YR : integer ) ;
      nakreslí úsečku z dosavadní polohy pisátka (X0,Y0) do bodu
      (X0+XR,Y0+YR), pisátko zůstane spuštěno

procedure QPNTR ( XR, YR : integer ) ;
      přesun (bez kresby) pisátka z dosavadní polohy (X0,Y0) do bodu
      (X0+XR,Y0+YR) a udělání tečky v tomto bodě, pisátko zůstane
      spuštěno
```

7.9.4. Tisk znaků

```
procedure QSIZE ( XX,XY,YX,YY : integer ) ;
      volba tvaru, sklonu a směru tisku

procedure QNARROW ;
      nastavení úzké mezery mezi znaky

procedure QWIDE ;
      nastavení široké mezery mezi znaky

procedure QWRITE ( <znakový řetězec> nebo char ) ;
      tiskne hodnotu parametru
```

7.9.5. Modifikace rychlosti kreslení

```
procedure QSPEED ( S : integer) ;
      přípustné hodnoty jsou 1, 2, 3, 4 a 5. Přípustnost hodnoty
      parametru se nekontroluje.
```

8. Práce se soubory

8.1. Parametry programu

V Pascalu pod OS AMOS nejsou dovoleny lokální soubory. Všechny soubory, se kterými pascalský program pracuje, musí tedy být jeho parametry (tzn. musí být uvedeny v hlavičce programu). Konkrétní fyzické soubory, které jim odpovídají, se specifikují ve volací větě, kterou se spouští běh programu nebo pomocí procedury ASSIGN (viz 8.8.).

Pokud ve volací větě nespecifikujeme některý ze souborů, bere se implicitní specifikace takto :

```
soubor    input          :CI:
ostatní soubory          :CO:
```

8.2. Iniciální stav souborů vstupujících z klávesnice

Podle standardu jazyka Pascal se má po otevření souboru F pro čtení procedurou RESET okamžitě do přístupové proměnné F načíst hodnota první položky souboru. To ovšem není dost dobře realizovatelné pro interaktivní soubory (program by čekal s otevřením souboru na vstup znaku z klávesnice), proto je tato situace u všech instalací Pascalu, pracujících s interaktivním vstupem, řešena jinak.

V OS AMOS jsou soubory, jimž je přiřazeno zařízení :CI: (jak už víme implicitně je to standardní vstupní soubor input), po otevření ve stavu EOLN = true a přístupová proměnná tedy obsahuje mezeru. čtení číselných údajů je možné bez jakýchkoli předchozích akcí. Chceme-li však číst znaky, je třeba nejprve provést příkaz READLN. Při psaní programu je tedy nutné počítat s tím, že v tomto případě se může lišit chování programu podle toho, zda se vstup uskutečňuje z interaktivního (:CI:) nebo neinteraktivního souboru.

Chceme-li tedy využívat interaktivní vstup do pascalského programu, můžeme postupovat např. takto:

```
write ( Zadej....) ;     {tisk výzvy}
readln ;                 {vyžádání vstupů jednoho řádku
                          z klávesnice do bufferu (musí
                          být odeslán klávesou CR )}
read ( ..... ) ;         {vlastní čtení z bufferu do
                          proměnných}
```

8.3. Interaktivní vstup čísel z klávesnice

Čte-li pascalský program ze vstupního textového souboru číslo a čtený obsah souboru neodpovídá syntaxi předepsané pro zápis čísla (např. začíná písmenem), pak dojde k běhové chybě se všemi jejími důsledky, tj. k ukončení výpočtu a výstupu post-mortem-dumpu.

Takové chovaní je v případě vstupu z obecného textového souboru jediné možné, v případě vstupu z klávesnice by však bylo pro uživatele velmi nepříjemné. Mohl by totiž pouhým překlepem přijít o celý předchozí výpočet. Proto je tato situace v OS AMOS řešena jinak.

Dojde-li při zpracování čísla vstupujícího z klávesnice k chybě, vypíše se na obrazovce chybové hlášení a zobrazí se znovu část naposledy odesílaného vstupního bufferu, počínaje prvním znakem čísla, které již nebylo přečteno, protože v něm byla odhalena chyba. Uživatel má možnost obvyklým způsobem tuto chybu ve vstupním bufferu opravit a odeslat jeho nový obsah. Nedopustil-li se znovu chyby, výpočet pokračuje dál, je-li vstup chybný, situace se opakuje.

8.4. Implicitní formáty výstupu

Při výstupu do textových souborů pomocí procedur WRITE a WRITELN bez udání formátu se použijí následující implicitní hodnoty formátu v závislosti na typu vystupující hodnoty

typ	počet znaků
integer	8
real	16
char	1
boolean	4 pro true, 5 pro false
řetězec	délka řetězce

8.5. Procedura REWRITE

Podle standardu jazyka je zavolání procedury REWRITE na soubor fakticky ekvivalentní jeho smazání. Kdyby tato procedura byla realizována voláním systémové akce DELETE, tj. faktickým smazáním souboru v paměti nebo na disku, mohl by si tak uživatel překlepem smazat cenný soubor. Kdyby se naopak soubor při zavolání procedury REWRITE nemazal, pak by nešlo tuto proceduru zavolat v programu na tentýž program vícekrát. Proto bylo zvoleno toto řešení:

Při prvém zavolání procedury REWRITE v programu nesmí soubor existovat, jinak dojde k chybě.

Při druhém a dalších voláních této procedury na soubor se již jeho obsah skutečně maže.

8.6. Otevření souboru pro přípis

Ve standardním Pascalu lze soubor otevřít pro zápis pouze pomocí standardní procedury REWRITE, to však zničí jeho předchozí obsah.

OS AMOS umožnil realizovat jako rozšíření standardního jazyka také proceduru

```
procedure EXTEND (var F: <typ soubor> ) ;
```

která otevře soubor pro přípis, tj. nastaví pozici souboru za konec aktuálního obsahu a povolí zápis do něj. Je tedy možné po otevření procedurou EXTEND soubor pomocí procedur PUT a WRITE "prodlužovat".

8.7. Uzavírání souboru

Přímo z Pascalských programů lze volat systémovou akci uzavírání souboru. Slouží k tomu procedura

```
procedure CLOSE( var F: <typ soubor>) ;
```

8.8. Procedure ASSIGN

```
procedure ASSIGN( var F: <typ soubor> ; S : <řetězec> ) ;
```

Pomocí této procedury lze uvnitř pascalského programu dynamicky přiřadit konkrétní fyzický soubor proměnné F typu soubor. Skutečným parametrem odpovídajícím parametru S může být libovolný řetězec, předpokládá se, že obsahuje specifikaci souboru.

Soubor pak lze otevřít a normálně s ním pracovat.

Např.

```
ASSIGN ( DISK , 'D0:SOUB.HEX' ) ;
reset (DISK) ;
```

8.9. Soubory s přímým přístupem

Po nahrání modulu s procedurami přímého přístupu k souborům do paměti RAM může programátor pracovat se soubory s přímým přístupem i z Pascalu.

K přímému přístupu je možné otvírat pouze datové soubory, tj. soubory deklarované v programu pomocí definice

```
file of ....  ,
```

(nikoli tedy textové soubory deklarované jako typ text).

Soubor OS AMOS, se kterým chceme pracovat v přímém přístupu musí být paměťový nebo diskový. V přímém přístupu lze pracovat pouze s hexadecimálními soubory.

Při přímém přístupu k souboru lze pomocí procedury SEEK nastavit pozici souboru na jeho požadovanou složku, kterou specifikujeme jejím pořadovým číslem, které se počítá od nuly. Na tuto pozici pak můžeme pomocí procedury PUT zapsat nový obsah, resp. její obsah přečíst procedurou GET. Obě tyto procedury současně posunou pozici u souboru o jednu dál.

I když tomu tak ve skutečnosti není, je výhodné si představovat, že každý soubor je ukončen speciální "ukončovací" složkou EOF. Pořadové číslo této složky lze zjistit pomocí procedury LASTPOS.

Pozici souboru nelze pomocí procedury SEEK nastavit za "ukončovací" složku souboru EOF.

8.9.1. Otevření souboru k přímému přístupu

```
procedure DIRECT ( var F : <datový soubor > ) ;
```

Otevřít k přímému přístupu lze jen datové soubory.

8.9.2. Nastavení pozice v souboru

```
procedure SEEK ( var F : <datový soubor> ; N : integer ) ;
```

Procedura nastaví aktuální pozici souboru na jeho N-tou složku. Soubor F musí být otevřen pro přímý přístup. Pokud by hodnota parametru N byla záporná nebo by specifikovaná pozice ležela již mimo soubor, je hlášena běhová chyba.

8.9.3. Zjištění aktuální pozice souboru

```
function POS ( var F : <datový soubor> ) : integer ;
```

Funkce vrací číslo aktuální pozice souboru F. Soubor F musí být otevřen pro přímý přístup.

8.9.4. Zjištění koncové pozice souboru

```
function LASTPOS ( var F : <datový soubor> ) : integer ;
```

Funkce vrací číslo pozice "ukončovací" složky EOF souboru F. Soubor F musí být otevřen pro přímý přístup.

9. Zvláštní rysy implementace

9.1 Rozsah platnosti identifikátoru

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

Např.

```
const I=5;
procedure A;
  type P= array [1..I] of char;
  var I: integer;
....
```

Tento úsek programu, je akceptován i když v bloku procedury A má identifikátor I dva různé významy, což není podle normy povoleno.

9.2. Alokace proměnných

Paměť je lokálním proměnným procedur přidělena staticky. Je tedy každá proměnná při každém volání procedury, v níž je lokální, alokována na stejné místo v paměti. Tím se na procesoru 8080, který nemá bázování ani indexování, dosahuje kratšího a rychlejšího kódu.

Při rekurzivním volání procedury se její lokální proměnné odsunou na zásobník a po ukončení procedury se 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 parametrů volaných referencí přistupovat ke starším exemplářům lokálních proměnných rekurzivních procedur.

9.3 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.

Např.

```
      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.

16. Řízení překladu

16.1. Přepínače

Práci překladače lze ovlivnit nastavením 7 přepínačů. každý přepínač je označen jedním písmenem a jeho funkce může být aktivována nebo potlačena. Přehled přepínačů je v následující tabulce:

jméno	význam	implic. nastavení
A	Generování testu při dosazování do proměnné typu interval	+
I	Generování testu při indexaci pole	+
P	Generování testu při vyhodnocování proměnné identifikované ukazatelem.	+
V	Generování testu nedefinované hodnoty proměnné	+
L	Vypisování protokolu o překladu (řádky s chybami nebo varováními se vypíší vždy)	+
S	Hlášení o použití rozšíření Pascalu oproti standardu	-
W	Varování při použití nebezpečné konstrukce	+

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

Implicitní nastavení přepínačů lze změnit uvedením parametru OPTIONS při volání překladače z OS AMOS (viz 15.3).

Nastavení přepínačů lze měnit kdykoliv v programu pomocí tzv. dolarových komentářů tímto zápisem:

```
(*$X+,Y-,...*)
```

kde X,Y jsou jména přepínačů, "+" způsobuje aktivaci a "-" potlačení funkce přepínače.

10.2. Řízení tisku protokolu o překladu

Do zdrojového programu lze umístit řádky, začínající znakem dolar, které nejsou součástí programu v Pascalu a slouží pouze k řízení tisku protokolu o překladu. Na těchto řádcích mohou být umístěny tyto příkazy:

$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 přechody na novou stránku, dokud se nenarazí na příkaz TITLE nebo EJECT

Nadpis je řetězec dlouhý 32 znaků, případné další znaky se ignorují. Počáteční hodnota nadpisu je 32 mezer (je prázdný). Pouhé zrušení nadpisu při zachování přechodů na novou stránku se provede příkazem CHANGE s prázdným nadpisem.

11. 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 prvém sloupci protokolu se nachází pořadové číslo zdrojové řádky. Započítávají se i řádky s příkazy pro řízení tisku protokolu (viz 10.2.).

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 každé hlavičky procedury, u BEGIN a END vymezujících blok. Pro hlavní program se tiskne "A", pro proceduru deklarovanou v hlavním programu "B" atd.

Dále následuje přesný opis zdrojové řádky.

Význam druhého sloupce se liší podle toho, zda na řádce vpravo jsou příkazy nebo deklarace.

Je-li na řádce příkaz (nebo jeho část), udává číslo hexadecimální adresu, od níž se příkaz začal ukládat do paměti. Např. podmíněny příkaz v příkladu se uložil od adresy 0B21H.

Jsou-li na řádce deklarace proměnných nebo parametrů procedury, je uvedena adresa, která je již obsazena před začátkem zpracování seznamu proměnných. Jednotlivé proměnné se postupně alokují pod touto adresou. Jinak řečeno číslo je o jedničku vyšší než je adresa posledního bytu přiděleného proměnným právě zpracovávaného seznamu. Podle konvencí pro přidělování paměti proměnným (viz. paragraf 4.) je to poslední byte poslední proměnné tohoto seznamu (která se z proměnných v seznamu alokuje jako první).

Podle údajů o paměťových nárocích jednotlivých typů pak již lze jednoduše vypočítat, na jaké adrese je daná proměnná alokována.

Ve výše uvedeném příkladu je:

```
parametr A na adrese 2F80H = 2FB2H - 2
proměnná R na adrese 2F7CH = 2F80H - 4
proměnná Y na adrese 2F7BH = 2F7CH - 1
proměnná X na adrese 2F7AH = 2F7BH - 1
```

12. 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.

Seznam chyb při překladu programu je v paragrafu 16.

13. Chyby při běhu programu

Množství běhových kontrol prováděných v Pascalu může pomocí běhových chyb zabránit nedefinovanému průběhu výpočtu. Proto je rušení běhových kontrol (viz 10.1.) vhodné až u odladěných programů a při zvláštních požadavcích na rychlost vypočtu nebo délku kódu.

Běhová chyba vždy způsobí ukončení výpočtu.

Překladač nehlásí jako chybu přístup k položce neaktuální větve proměnné typu větvený záznam.

Po běhové chybě se na obrazovce objeví nápis:

```
Běhová chyba, tisk dumpu na tiskárnu? (A/N)
```

Stiskneme-li klávesu "A", vypíší se na připojenou tiskárnu informace o stavu výpočtu programu v okamžiku, kdy k chybě došlo, tzv. post-mortem-dump. Stiskneme-li klávesu "N" (nebo jinou klávesu), vypisují se tyto informace na obrazovku.

Příklad takového výpisu následuje:

```
*** CHYBA *** 63
PROGRAM UKONCEN NA ADRESE 0283
  V BLOKU FIBO LOKALNI PROMENNE (HEX):
2FE1:DC DC DC DC DC DC DC DC DC DC DC DC DC DC DC
2FF0: DC DC DC DC DC 00 06 30 31
VOLANI NA ADRESE 02C9
  V HLAVNIM PROGRAMU LOKALNI PROMENNE (HEX):
2FF9: DC DC DC DC 01 00

HALDA     OD: 02E0  DO:02E0
ZASOBNIK  OD: 2FE0  DO:2FD2
```

Na první řádce chybového hlášení je číslo chyby 63. Z přehledu běhových chyb zjistíme, že došlo k dělení nulou v operaci DIV nebo MOD.

Chyba nastala při provádění procedury nebo funkce FIBO na adrese 0283H. 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.

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 02C9H, následují tedy proměnné deklarované v hlavním programu.

Hlášení je ukončeno informací o adresovém prostoru, v němž je halda a zásobník. Obsah těchto buněk lze zjistit odskočením do monitoru.

Výstup chybového hlášení na obrazovku se po jejím zaplnění sám pozastaví.

Pak je možné pokračovat třemi způsoby :

- stiskem klávesy E výpis ukončit,
- stiskem klávesy N přejít rovnou k výpisu informací o další proceduře z pracovního řetězce
- stiskem jiné klávesy (např. CR) nechat vystoupit další obrazovku chybového hlášení

Seznam běhových chyb je v paragrafu 17.

14. Spuštění překladače Pascal z OS AMOS

Příkaz pro spuštění překladače Pascalu má tvar

PAScal [f1] [,[f2] [,f3]] [;<params>] kde

f1 je specifikace souboru obsahujícího zdrojový program.

f2 je specifikace souboru, do kterého se bude vytvářet object-kód. Není-li specifikován, vytvoří se soubor stejného jména jako byl f1 s implicitní předponou a příponou OBJ. Specifikuj-li uživatel jinou příponu než OBJ, je tato násilně přepsána na OBJ.

f3 specifikace souboru, do kterého se vytváří listing. Není-li specifikován, bere se implicitní hodnota :CO:, tj. výstup listingu na obrazovku.

<params> je řetězec obsahující parametry překladu.

14.1. Tvar řetězce <params>

Řetězec může obsahovat specifikaci parametrů překladu. Všechny parametry jsou klíčové, žádný není povinný. Specifikace jednotlivých parametrů jsou od sebe odděleny jednou nebo více mezerami.

14.2. Parametry ovlivňující hospodaření s pamětí

O(číslo) číslo udává velikost paměti v sektorech, které se rezervují na vygenerovaný kód programu a jeho data. Implicitní hodnota je 16.

Z(číslo) číslo udává velikost paměti v sektorech, které se rezervují pro vytváření listingu do paměti. Implicitní hodnota je 0.

Zbytek paměti (tj. paměť, která není obsazena soubory a nebyla parametrem Z vyhrazena pro listing) je používán překladačem jako pracovní prostor.

Potřebujeme-li přesně vymezit oblast, kam se bude umísťovat přeložený program, můžeme to udělat pomocí parametrů ZAC a KON.

ZAC(číslo) adresa prvního bytu, do kterého se umístí kód programu

KON(číslo) adresa prvního bytu, za oblastí pro přeložený program

14.3. Inicializace přepínačů překladu

Pomocí parametru OPTIONS můžeme změnit iniciální hodnoty přepínačů programu (viz 11.1)

Např.

OPTIONS(L-,S+) potlačí vytváření listingu a vyvolá hlášení varování při použití rozšíření oproti standardnímu Pascalu.

15. Spuštění přeloženého pascalského programu

Tvar příkazu spouštějícího uživatelský program byl popsán v kapitole II. paragraf 3.2.

Je-li např. v souboru ALFA.OBJ object-kód přeloženého pascalského programu, spustíme jeho výpočet příkazem

ALFA [<spec. souborů>] [;<parms>]

Příponu OBJ tedy není nutné uvádět. OS AMOS nejprve tento object-kód naalokuje a pak přečte <spec. souborů>. V tomto seznamu musejí být specifikovány soubory, se kterými program má pracovat. Specifikace musí být ve stejném pořadí, v jakém jsou uvedeny identifikátory jim odpovídajících souborů v seznamu parametrů pascalského programu. Specifikace jednotlivých souborů se oddělují čárkami.

Specifikace standardních souborů INPUT a OUTPUT mohou chybět. V tom případě se berou implicitní hodnoty

:CI: pro INPUT
:CO: pro OUTPUT

Např.

Je-li v souboru ALFA.OBJ object-kód programu s hlavičkou

```
PROGRAM A (INPUT,OUTPUT)
```

můžeme jeho výpočet spustit příkazem

```
ALFA
```

V tomto případě bude vstup z klávesnice a výstup na obrazovku.

Spustíme-li výpočet příkazem

```
ALFA :MC:VSTUP,VYSTUP
```

bude se soubor INPUT číst z magnetofonového souboru se jménem VSTUP a výstupní soubor OUTPUT se bude vytvářet do paměti, kde bude později přístupný pod jménem VYSTUP.

16. Seznam chyb při překladu pascalských programů

16.1. 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 nedovede vzpamatovat z chyb uživatele

5 příliš rozsáhlé deklarace

6 příliš dlouhý program

7 vnitřní chyba překladače

8 nedostatečná volná paměť pro překladač

16.2. Syntaktické chyby

1 nepřípustný znak ve zdrojovém textu

2 vstupní řádka je příliš dlouhá

3 znak "_" se nesmí vyskytovat uvnitř vyhrazeného slova

4 znakový řetězec nesmí přesáhnout řádku

5 prázdný znakový řetězec je nepřípustný

6 za identifikátorem konstanty nebo typu je očekáván

7 příliš velká integerová konstanta nebo celá část reálné konstanty

8 konstanta začíná nepřípustným symbolem

9 použití lokálních souborů není implementováno

10 nepřípustný symbol

11 očekáván znak "," nebo ")"

12 tělo bloku musí začínat vyhrazeným slovem BEGIN

13 očekáván znak ":" a identifikátor typu funkce

14 očekáván identifikátor konstanty

15 skutečným parametrem, volaným referencí musí být proměnná

16 očekáván identifikátor typu

17 faktor začíná nepřípustným symbolem

18 procedury READ a WRITE musejí 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 očekáván symbol ".."

26 očekáván znak "," nebo "]"

28 očekáván "PROGRAM"

32 očekávána tečka

45 očekávána levá závorka

46 očekávána čárka

47 očekávána celočíselná konstanta

48 očekávána pravá závorka

49 očekáváno ":="

50 očekáván středník

51 očekávána dvojtečka

52 očekáváno "THEN"

55 očekáváno "END"

56 očekáváno "UNTIL"

58 očekáváno "OF"

59 čekáváno "DO"

60 očekáváno "TO" nebo "DOWNTO"

69 očekáván 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 BOOLEAN/CHAR/řetězec/výčtový typ nelze použít operací "+", "-", "*", "/";
pro množiny je nepřípustná operace "/"

76 argument funkce/procedury musí být typu REAL nebo INTEGER

77 argument funkce/procedury musí být typu INTEGER

78 argument procedur NEW, DISPOSE, MARK, RELEASE musí být typu ukazatel

79 argument procedury/funkce musí být typu soubor, argument EOLN, PAGE musí být typu text

80 procedury READLN, WRITELN lze použít pouze na textový soubor

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 nebo znakový 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 množiny musí ležet v intervalu 0 až 255

90 typ funkce nesmí být strukturovaný

91 očekáván 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 očekávána konstanta typu INTEGER

111 volání procedury nesmí být součástí výrazu

112 očekávána proměnná

113 řídící 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 identifikátor 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

126 v zápisu reálné konstanty musí po desetinné tečce následovat číslice

127 v exponentu reálné konstanty musí být alespoň jedna číslice

128 příliš velká reálná konstanta

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

140 příliš velký rozsah proměnných

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

160 Druhý skutečný parametr procedury ASSIGN musí být typu znakový řetězec

161 První parametr procedury/funkce musí být datový soubor, tj. soubor definovaný pomocí FILE OF ...

162 Parametr procedury QWRITE musí být typu CHAR nebo znakový řetězec

163 Parametr funkce KEY není typu CHAR

Kromě hlášení chyb překladač v některých případech vydává varování:

201 test na rovnost nebo nerovnost s operandy typu REAL může být nevhodný

202 v komentáři se vyskytl symbol ";" nebo "(*"

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

246 použití standardní procedury nebo funkce, která není součástí standardního Pascalu

247 návěští má více než 4 cifry

248 změna typu

250 operace umocňování

17. Seznam chyb při běhu pascalských programů

0 přerušení programu klávesou BREAK - nejde o chybu, ale o zásah obsluhy do běhu programu. Dojde k výpisu post-mortem-dumpu a k návratu do CCP AMOS.

1 pokus o zápis do nestandardního souboru před provedením REWRITE
(nastává při u příkazů WRITELN a PUT)

2 pokus o čtení z nestandardního souboru před provedením RESET
(nastává u příkazů READ, READLN a GET)

3 pokus o čtení ze souboru po přečtení všech jeho složek (platí podmínka EOF)
(nastává u příkazů READ, READLN a 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 hostitelského typu integer
(dosazování do proměnné typu interval, indexace)

7 překročení horní meze typu interval vytvořeného z hostitelského typu integer
(dosazování do proměnné typu interval, indexace)

8 na vstupu není číslice
(nastává u příkazů READ, READLN pro argument typu integer nebo real)

9 na vstupu je příliš velké číslo
(nastává u příkazů READ a READLN)

11 pokus o otevření souboru, který je již otevřen

14 ve výrazu byla použita proměnná s nedefinovanou hodnotou

15 chybný parametr procedury DISPOSE

16 vyčerpání prostoru pro dynamickou alokaci proměnných
(nastává u příkazu 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 (*,/)

30 příliš velké reálné číslo na vstupu
(nastává u příkazů READ a READLN pro argument typu REAL)

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

33 menší hodnota v tomto typu neexistuje (PRED)

34 větší hodnota v tomto typu neexistuje (SUCC)

35 aktuální hodnotě selektoru v CASE příkazu neodpovídá žádná z jeho větví

36 změna hodnoty řídící proměnné cyklu v jeho těle

47 hodnota nemůže být prvkem množiny
(nastává při zpracování konstruktoru množiny)

48 překročení mezí bázového typu typu množina
(nastává při dosazování množin)

50 vyčerpání paměti
(rekurzivní volání procedur a funkcí)

61 přetečení nebo podtečení v aritmetických operacích pro typ INTEGER (+,-,*,SQR)

62 exponent musí být nezáporný (**)

63 dělení celého čísla nulou (DIV, MOD)

64 druhý operand v operaci MOD je záporný (MOD)

80 příliš velký argument exponenciely (EXP)

81 argument logaritmu je 0 (LN)

82 argument logaritmu je záporný (LN)

83 argument odmocniny je záporný (SQRT)

84 pozice s daným pořadovým číslem leží mimo soubor, toto číslo je záporné nebo příliš velké
(nastává u procedury SEEK pro přímý přístup)

85 soubor nebyl otevřen pro přímý přístup

88 nepřípustný formát pro výstup reálné hodnoty
(WRITE, WRITELN)

91 hodnoty skutečných parametrů procedury KURZOR zadávají polohu mimo obrazovku

99 byl proveden příkaz HALT, nejde o chybu,
ale o úmyslné ukončení běhu programu požadované programátorem