Vysoká škola strojní a elektrotechnická v Plzni

Katedra technické kybernetiky

Ing. Václav Skala

Příspěvek k filosofii implementace relace v relační bázi dat

Kandidátská disertační práce

Školitel: Prof. Ing. Stanislav Kubík, DrSc.,
člen korespondent ČSAV

Plzeň, 1978

Je mojí milou povinností poděkovat mému školiteli Prof. Ing. Stanislavu Kubíkovi, DrSc., členu korespondentu ČSAV za vedení a všestrannou podporu při práci.

1. Úvod

Relační modely byly při zpracování dat poprvé použity v roce 1964 jako součást projektu, který měl umožnit, aby počítač odpovídal na dotazy související s textem uloženým v jeho paměti v hovorovém jazyce. Podstata využívání relačních modelů je ve specifickém využití relačních množin, které vyjadřují vztahy mezi několika dalšími množinami. Přestože se teorie relačních modelů zabývá spíše způsoby organizace dat než jejich skutečným uložením, jde o zcela neobvyklé a nepochybně velmi perspektivní pojetí, které bude mít pro uložení dat velký význam.

Z určitého hlediska jsou relační modely jen další metodou vyjádření uživatelova pojetí dat v bázi dat. V rámci relačního modelu jsou všechny soubory dat (nazývané relace) považovány za lineární řadu záznamů (nazývané n-tice), kde každý záznam obsahuje určité atributy (nazývané oblasti).

Smysl a nesporný perspektivní význam použití relačních modelů lze specifikovat dvěma body, které jsou současně dostatečným zdůvodněním pro další studium této metody :

1. Tato metoda poskytuje podstatně vyšší stupeň nezávislosti uživatelů báze dat vytvořením zcela abstraktního modelu dat, který lze využít k definování všech vlastností uvažovaných dat a na němž může tvůrce báze dat ověřit nejlepší způsob jejich uložení v bance dat.
2. Tato metoda poskytuje možnost využít ověřený matematický aparát pro práci s množinami na relacích báze dat, které mohou využívat takto formulované teoretické modely. Metodu relačních modelů rozvíjí v poslední době zejména firma IBM, jako alternativu k návrhu systému banky dat CODASYL.

2. Formulace problému

2.1. Srovnání současných databázových systémů

Relační databázové systémy, které byly implementovány asi do roku 1970, byly experimentálního typu a nebyly běžně přístupné, ačkoliv některé dostupné systémy měly již určité relační prvky např. ADABAS [56] . Mezi první relační databázové systémy patří : LEAP, TRAMP, Relational Data Files, které poskytují pouze možnost pracovat s binárními relacemi a systém STDS, který umožnil pracovat s relacemi libovolného stupně.

Uveřejnění původního článku E.F.Codda [9] v roce 1969 dalo podnět ke zvýšení aktivity v oblasti relačních databázových systémů. Tak vznikly např. systémy : Query by Example, SQUARE, SEQUEL, System R, SQUIRAL, DAMAS, INGRES, RENDEZVOUS, MacAIMS, ZETA, GAMMA-0, MORIS, CASSM, RDMS.

Nyní budou uvedeny velmi stručně charakteristiky systémů, které jsou autorovi známy.

CASSM (Context-Adressed Segment-Sequental Memory) předkládá návrh pro přímou podporu relací a operací s relacemi hardwarem. Poskytované datové struktury a operace jsou podobné těm, které jsou poskytovány běžnými systémy na vnějších pamětích. Předpokládá se implementace jazyka SLICK, který je podobný SEQUELu.

GAMMA-0 je hypotetický interface, který by měl být použit při implementaci např. relačního kalkulu. Systém GAMMA-0 nebyl nikdy implementován, avšak mnoho principů bylo zahrnuto do RSI (Relational Storage Interface) v systému R.

ZETA je systém implementovaný v Torontu. Pracuje s jednoduchými n-ticemi. Umožňuje spolupráci s programy napsanými v jazyce PL/I, kde jsou přístupné příkazy podobné příkazům SEQUELu a používání interfacu TORUS pro uživa-

telské dotazy v přirozeném jazyce.

MORIS (Modular Relational Information System) poskytuje možnosti různé organizace relací (např. stromovou, seznamovou). Umožňuje používání nenormalizovaných relací. Pro používání tohoto systému byl implementován jazyk COLARD, který umožňuje jak algebraické operace, tak i relační kalkul.

IS/1 - realizovaný firmou IBM používá k uložení relací sekvenční organizace souboru. Uživateli jsou poskytovány algebraické operace. Z tohoto systému vznikly později systémy UMS a PRTV.

V tab.2.1.1. je přehled význačnějších relačních databázových systémů s uvedením stupně relací a orientací jazyka.

Systém	Relační model dat	Orientace na
LEAP	binární	jednotlivé n-tice
TRAMP	binární	jednotlivé n-tice
Relational Data Files	binární	relační kalkul
STDS	n-ární	algebraické operace
RDMS	n-ární	algebraické operace
Cambrige	binární	relační kalkul
MacAIMS	n-ární	algebraické operace
IS/1	n-ární	algebraické operace
SQUARE	n-ární	relační kalkul
SEQUEL	n-ární	relační kalkul
MORIS	n-ární (nenormalizovaný)	relační kalkul algebraické operace

Přehled význačnějších relačních bází dat
Tab.2.1.1.

2.2. Formulace problému

Z dostupných informací vyplývá, že skutečně realizované relační databázové systémy vyžadují pojem klíče, a to především z důvodů jednoznačné identifikace n-tice v relaci. To má za následek rychlý výběr n-tic, je-li znám jejich klíč, avšak téměř sekvenční prohledávání relace v případě, že klíč požadovaných n-tic znám není. Další nevýhodou je, že se hodnoty většinou pamatují přímo v n-tici dané relace. V případě, že jde o delší znakové řetězce, je spotřeba paměti neúnosně velká. Pouze některé systémy připouštějí možnost kódovat data, a to ještě některé z nich tak, že se předepíše přiřazení určitého kódu k určité hodnotě. V případě, že budeme akceptovat přístup založený na tom, že známe hodnotu klíče, lze relace uchovávat např. jako index - sekvenční soubory. Tento přístup není přijatelný, pokud budeme chtít vybírat n-tice relace i podle neklíčových položek, neboť doba odezvy na dotaz pak vzroste nad únosnou mez.

Cílem práce je navrhnout filosofii implementace relace relačního databázového systému, který je určen především pro velmi rychlé zodpovězení dotazů nad relacemi, jejichž n-tice jsou tvořeny dlouhými znakovými řetězci s možností automatického kódování hodnot bez zavedení pojmu klíče.

Při návrhu budeme uvažovat počítač, jež má následující vlastnosti:

a) virtuální paměť
b) implementovaný jazyk podobný jazyku Algol 68.

Práce si neklade za cíl navrhnout fyzický způsob implementace relace relačního databázového systému.

3. Definice relací a operace s relacemi

Budiž D_1, D_2,..., D_n množiny (nemusí být různé). Relace R definovaná na těchto množinách je definována jako jistá podmnožina kartézského součinu $D_1 x\ D_2 x \ldots x D_n$. Je tedy tvořena uspořádanými n-ticemi prvků tvaru $(e_1, e_2, \ldots, e_n)$, přičemž platí $e_i \in D_i$ pro $i = 1,2,\ldots,n$. Množinu D_i nazýváme i-tou oblastí relace R, resp. oblastí D_i. Číslo n nazýváme stupněm relace. Relace stupně 1 označujeme jako unární, relace stupně 2 jako binární, relace stupně n n-ární a používáme značení $R(D_1, D_2, \ldots, D_n)$.

K představě relace R lze použít pole, které vyhovuje vlastnostem:

a) každý řádek vyjadřuje jednu n-tici obsaženou v R
b) všechny řádky jsou navzájem různé,
c) uspořádání řádků je nevýznamné,
d) uspořádání sloupců odpovídá pořadí oblastí relace R,
e) sloupce jsou označeny jménem příslušné oblasti.

Prvky množiny $D_i (i = 1,2, \ldots,n)$ mohou být:

a) číselné konstanty,
b) znakové (resp. řetězové) konstanty,
c) relace definované nad určitými množinami $D_1', D_2', \ldots, D_k'$ (obecně $k \neq n$).

V případě ad a) a ad b) mluvíme o jednoduché oblasti, v případě ad c) mluvíme o složené oblasti.

Poznámka 1

Někdy, pokud to bude nutné, bude označována oblast jako položka, resp. n-tice jako věta, abychom se vyhnuli nejasnostem.

Relace, jejíž oblasti jsou znakové nebo číselné hodnoty, si lze představit např. takto:

ZAMĚSTNANEC:

JMÉNO	MZDA	VEDOUCÍ	ODDĚLENÍ
NOVÁK	2700	MATLÁK	HRAČKY
KŘIVAN	2100	MATLÁK	HRAČKY
ČEJKA	2050	SÝKORA	NÁBYTEK

Relace, jejíž oblasti jsou n-ticemi jiné relace, si lze představit např. takto:

POPLATNÍK:

JMÉNO	POHLAVÍ	DĚTI			PŘÍJEM	DAŇ
		JMÉNO	POHLAVÍ	VĚK		
NOVÁK	MUŽ	JIŘÍ ANNA	MUŽ ŽENA	19 23	4500	260
KŘIVANOVÁ	ŽENA	KAREL KARLA	MUŽ ŽENA	3 14	2700	130
KŘIVAN	MUŽ	ALFRÉD	MUŽ	27	4000	240

Tento typ relace se někdy nazývá relací v nenormalizovaném tvaru.

Pod pojmem báze dat nyní budeme chápat množinu relací. Tento přístup umožňuje aplikovat takové operace s relacemi, které jsou známé z matematické teorie množin.

Pro jednoduchost budeme zatím uvažovat relace, jejichž oblasti jsou jednoduché. Abychom mohli definovat množinové operace, musíme zavést pojem kompatibility relací.

Řekněme, že relace S $(S_1,\ldots,S_K)$ a $T(T_1,\ldots,T_n)$ jsou kompatibilní, jestliže platí:

1) $k = n$,

2) typ oblasti S_i je stejný jako typ oblasti T_i pro $i = 1,2,\ldots,n$,

kde pod typem oblasti rozumíme to, že oblast je buď číselná konstanta nebo znaková konstanta. Pro kompatibilní relace můžeme definovat operace sjednocení, průniku a rozdílu. Operace průniku a rozdílu neskrývají žádný problém.

Průnik

Uvažujme relace $R = (r_1,\ldots,r_n)$ a $S = (s_1,\ldots,s_n)$, které jsou kompatibilní. Pak průnik je definován:

$$R \cap S = \{ (t_1,\ldots,t_n)/(t_1,\ldots,t_n) \in R \wedge (t_1,\ldots,t_n) \in S \}$$

Rozdíl

Uvažujme dvě relace $R = (r_1,\ldots,r_n)$ a $S = (s_1,\ldots,s_n)$, které jsou kompatibilní. Pak rozdíl je definován:

$$R - S = \{ (t_1,\ldots,t_n)/(t_1,\ldots,t_n) \in R \wedge (t_1,\ldots,t_n) \notin S \} .$$

Podstatně složitější situace vzniká v případě sjednocování relací, protože musí být zajištěna vlastnost jedinečnosti n-tice v relaci.

Sjednocení

Uvažujme relace $R = (r_1,\ldots,r_n)$ a $S = (s_1,\ldots,s_n)$, které jsou kompatibilní. Pak sjednocení relací lze definovat:

$$R \cup S = \{(t_1,\ldots,t_n)/(t_1,\ldots,t_n) \in R \vee (t_1,\ldots,t_n) \in S\text{-}R^{*)}\}.$$

*) S-R slouží k tomu, abychom zdůraznili vyloučení stejných n-tic. Z hlediska teorie množin je možné psát jen S.

Poznámka

Je vhodné klást navíc požadavek na sjednocení relací, aby sémantická stránka operace sjednocení měla smysl. Nemělo by patrně smysl, aby byly sjednoceny dvě relace, z nichž první má v k-té oblasti údaje o mzdě a druhá má v k-té oblasti např. číslo zaměstnance. Po sjednocení by patrně k-tá oblast relace byla těžko použitelná.

Pro objasnění operací uveďme příklad.
Uvažujme dvě kompatibilní relace $R = (r_1,\ldots,r_n)$ a $S = (s_1,\ldots,s_n)$ pro $n = 2$.

R:

r_1	r_2
a	1
b	1
a	2
b	2

S:

s_1	s_2
a	1
b	2
c	1
e	4

Pak $T = (t_1, t_2)$ nabývá hodnot:

a) pro operaci průniku $T = R \cap S$:

t_1	t_2
a	1
b	2

b) pro operaci rozdílu T = R-S :

t_1	t_2
b	1
a	2

c) pro operaci sjednocení T = R ∪ S:

t_1	t_2
a	1
b	1
a	2
b	2
c	1
e	4

Kromě výše uvedených množinových operací existují takové operace, které umožňují vytvářet nové relace jiným způsobem.

Kartézský součin

Uvažujme dvě relace $R = (r_1,\ldots,r_n)$ a $S = (s_1,\ldots,s_K)$. Pak kartézským součinem $R * S$ relace R a relace S nazýváme relaci stupně n+k.

Tedy :

$$R * S = \{(r_1,\ldots,r_n, s_1,\ldots,s_K)/(r_1,\ldots,r_n) \in R \wedge (s_1,\ldots,s_K) \in S\}$$

Projekce

Uvažujme relaci $R = (r_1,\ldots,r_n)$. Pak projekcí relace R na oblasti $i_1, i_2,\ldots, i_K$ rozumíme relaci stupně k:

$$R\%\,(r_{i_1},\ldots,r_{i_K}) = \{(r_{i_1},\ldots,r_{i_K})/(r_1,\ldots,r_n) \in R\}$$

Přičemž platí : $i_j \in \langle 1,N\rangle \;\forall\; j \in \{1,2,\ldots,k\}$

Výběr

Uvažujme relaci $R = (r_1,\ldots,r_n)$. Pak výběrem podle kriteria F n-tic z relace R rozumíme relaci:

$$R:F = \{(r_1,\ldots,r_n)/(r_1,\ldots,r_n) \in R \wedge F(r_1,\ldots,r_n) = \text{true}\}$$

Θ - spojení

Uvažujme relační operátor Θ tj. ($\Theta \in \{=, <, \leq, >, \geq, \neq\}$). Uvažujme dále relace $R = (r_1,\ldots,r_n)$ a $S = (s_1,\ldots,s_K)$. Nechť oblasti r_i a s_j jsou stejného typu. Pak Θ -spojením relace R v r_i a relace S v s_j rozumíme relaci:

$$R\,[r_i \,\Theta\, s_j]\,S = \{(r_1,\ldots,r_n,s_1,\ldots,s_K)/\ (r_1,\ldots,r_n) \in R \wedge (s_1,\ldots,s_K) \in S \wedge (r_i \,\Theta\, s_j)\}$$

Další operace jsou:

a) ekvispojení - Θ operátor je operátor =

b) přirozené spojení - ekvispojení s vynecháním oblasti r_i resp. s_j

c) kompozice - ekvispojení s vynecháním oblasti r_i i s_j.

Pro objasnění operací uveďme příklad. Uvažujme dvě binární relace:

R:

r_1	r_2
Z	4
X	5
W	0

S:

s_1	s_2
A	9
B	8
W	0

Pak relace nabývá hodnot:

a)

$T = R * S$:

t_1	t_2	t_3	t_4
Z	4	A	9
Z	4	B	8
Z	4	W	0
X	5	A	9
X	5	B	8
X	5	W	0
W	0	A	9
W	0	B	8
W	0	W	0

b) pro operaci projekce

$U = T\ \%\ (t_2)$:

u_1
4
5
0

c) pro operaci výběru

$Q = T:\ ((t_2 = 0 \vee t_4 = 0) \wedge (t_1 = \text{'W'} \vee t_3 = \text{'W'}))$

Q :

q_1	q_2	q_3	q_4
W	0	W	0
W	0	A	9
W	0	B	8
X	5	W	0
Z	4	W	0

d) pro operaci θ -spojení

$P = R[r_1 = s_1]\ s$:

p_1	p_2	p_3	p_4
W	0	W	0

$P = R[r_2 < s_2] S$:

p_1	p_2	p_3	p_4
Z	4	A	9
Z	4	B	8
X	5	A	9
X	5	B	8
W	0	A	9
W	0	B	8

Výše uvedené operace je možné definovat i pro nejednoduché oblasti relace, tj.,pro relace v nenormalizovaném tvaru např. v [3].

4. Jazyk pro relační bázi dat

Od zavedení relačního modelu E.F.Coddem bylo navrženo několik jazyků určených pro uživatele - nespecialisty např. QUEL, Query by Example, SEQUEL viz. např.[13]. Jeden takový jazyk je jazyk SEQUEL, který je založen na anglických klíčových slovech a je vhodný pro použití nespecialisty ve zpracování dat právě tak dobře jako profesionálními programátory. Byla provedena série testů, v kterých byly dotazové schopnosti SEQUELu předkládány studentům se zkušenostmi v programování i bez nich [46]. Tyto testy oddělily několik rysů jazyka SEQUEL, které byly zdrojem obtíží při učení. Proto bylo provedeno několik změn v dotazovacích prvcích jazyka SEQUEL, který byl navíc rozšířen pro manipulaci s daty o vybavení, které připouští vkládání, rušení a změnu jednotlivých n-tic nebo množin n-tic, pro definici dat o vybavení, které připouští definici relací a různých pohledů na relace, pro řízení dat o vybavením které umožňuje každému uživateli povolit ostatním uživatelům přistupovat k jeho datům a které zajišťuje integritu dat.

Výsledkem těchto rozšíření a zlepšení je jazyk SEQUEL 2 [1], jehož vlastnosti budou stručně popsány. Pracuje s relacemi v první nebo vyšší normální formě, jak byly popsány Coddem [11]. Vlastnost jazyka SEQUEL 2 budou ilustrovány na příkladech:

Uvažujme bázi dat, která obsahuje následující čtyři relace:

ZAM = (Č-ZAM,JMÉNO,Č-ODDZ,PRÁCE,VEDOUCÍ,MZDA,PRÉMIE)

kde: Č-ZAM udává osobní číslo zaměstnance,
JMÉNO udává jméno zaměstnance,
Č-ODDZ udává číslo oddělení,
PRÁCE udává pracovní zařazení,
VEDOUCÍ udává osobní číslo vedoucího
MZDA udává mzda zaměstnance,
PRÉMIE udává prémii zaměstnance.

ODD = (Č-ODD,JMÉNO-ODD,MÍSTO)
kde : Č-ODD je číslo oddělení,
JMÉNO-ODD je název oddělení,
MÍSTO je umístění oddělení.

POUŽITÝ = (Č-ODDA,ČÁST),
kde: Č-ODDA je číslo oddělení používající část ČÁST.

DODÁVKA = (DOD,ČÁST A)
kde: DOD je dodavatel dodávající části ČÁST A.

4.1. Prostředky jazyka pro dotazy

Nejzákladnější operace jazyka SEQUEL 2 se nazývá mapování a je ukázána v př.1. Mapování říká, že známá hodnota (Č-ODD=50) má být transformována do požadované hodnoty JMÉNO pomocí relace ZAM. Mají se vrátit hodnoty oblastí, jež byly uvedeny v seznamu doložky select. Má-li se vrátit celá n-tice, lze psát select*. Doložka where může obsahovat libovolnou posloupnost predikátů, které přirovnávají atributy n-tic k hodnotám (např.Č-ODD=50) nebo srovnávají dva atributy n-tice navzájem (MZDA ≥ PRÉMIE). Predikáty mohou být spojeny pomocí AND a OR a závorky mohou být použity k vyjádření precedence.

Příklad 1

Vyber jména zaměstnanců v oddělení č.50.

select JMÉNO from ZAM where Č-ODDZ=50

Obecně mapování vrací množinu hodnot, které vyhovují doložce where vybraných atributů n-tice. Duplicitní hodnoty ve vrácené množině nejsou eliminovány, ledaže by uživatel napsal:

select unique JMÉNO from ZAM where Č-ODDZ=50

Rozhodnutí, zda se bude vyžadovat **jedinečnost**, bylo svěřeno uživateli, neboť vyloučení duplicitních hodnot je příliš drahé.

Další příklad ukazuje projekci relace ZAM na atribut Č-ODDZ.

Příklad 2

Vytvoř seznam všech různých čísel oddělení v relaci ZAM.

```
select unique Č-ODDZ from ZAM
```

Predikát v doložce where může také testovat atributy uložené v množině, jak ukazuje příklad 3.

Příklad 3

Udělej seznam jmen zaměstnanců z oddělení 25,47,33.

```
select JMÉNO from ZAM where Č-ODDZ in (25,47,33)
```

Je však také možno použít výsledek mapování v doložce where jiném mapování. Takové mapování se nazývá hnízdové a může být "zanořeno" do libovolné úrovně.

Příklad 4

Nalezni jména zaměstnanců, kteří pracují v oddělení, které je umístěno v Plzni.

```
select JMÉNO from ZAM where Č-ODDZ in
        select Č-ODDZ from ODD where MÍSTO='PLZEŇ'
```

SEQUEL 2 vyžaduje uvozovky omezující všechny znakové řetězce (jako hodnoty), aby se tím odlišily od jmen atributů. Uvozovky jsou volitelné u numerických konstant.

Pořadí vrácených hodnot je určeno systémem. Toto pořadí může uživatel změnit, jak ukazuje příklad 5., specifikací hlavního třídicího atributu a vedlejších třídících atributů.

Příklad 5

Vyber seznam čísel zaměstnanců, jejich jmen a jejich platů v oddělení č.50 v pořadí podle čísla zaměstnance.

```
select Č-ZAM, JMÉNO,MZDA from ZAM where Č-ODDZ=50
                                order by Č-ZAM
```

SEQUEL 2 poskytuje též uživateli možnost používat vestavěné funkce např. COUNT,AVG,MIN,MAX,SUM nebo možnost definovat nové.

Příklad 6

Nalezni průměr mzdy úředníků.

```
select AVG (MZDA) from ZAM where PRÁCE= 'UŘEDNÍK'
```

Označení COUNT (*) označuje načítávání n-tic,které vyhovují doložce where.

SEQUEL 2 uvažuje u relací s možností neznámých hodnot, které jsou ignorovány při použití vestavěných funkcí vyjma funkce COUNT. Při určování , zda daná n-tice vyhovuje doložce where v dotazu, se používá tříhodnotová logika viz obr.4.1. Je-li hodnota doložky where TRUE, pak nalezená n-tice se považuje za vyhovující, v případě že doložka where nabývá hodnoty ? nebo FALSE za nevyhovující.

AND	T	F	?
T	T	F	?
F	F	F	F
?	?	F	?

OR	T	F	?
T	T	T	T
F	T	F	?
?	T	?	?

NOT	
T	F
F	T
?	?

IF X THEN Y	X=T	Y=F	Y=?
X = T	T	F	?
X = F	T	T	T
X = ?	T	?	?

obr.4.1.

Výjimka z výše uvedených pravidel je udělána v případě tvrzení, které hledá neurčené hodnoty, tj. doložka where má např. tvar: where MZDA = NULL. V tomto případě je s neurčenou hodnotou zacházeno jako s ostatními hodnotami.

4.2. Možnosti jazyka pro manipulaci s daty

Prostředky pro manipulaci s daty umožňují uživateli měnit hodnoty přímo v bázi dat. Jsou poskytovány operace vkládání a rušení n-tic a operace pro změnu hodnot oblastí v n-tici.

Vkládání n-tice do relace je ukázáno v příkladě 1 s tím, že všechny hodnoty nejsou určeny. V případě, že jsou určeny všechny hodnoty, lze vynechat seznam oblastí relace, do níž hodnoty vkládáme.

Příklad 1.

Vlož nového zaměstnance, který se jmenuje NOVÁK s osobním číslem 553 a který pracuje v oddělení č.151. Ostatní hodnoty nejsou zatím specifikovány.

insert into ZAM (Č-ZAM,JMÉNO,Č-ODD-Z): <553, 'NOVÁK', 151>

Tomuto zápisu je ekvivalentní zápis:

insert into ZAM: <553,'NOVÁK', 151,NULL,NULL,NULL,NULL>

Příkaz vkládání lze kombinovat s dotazem na nějakou již existující relaci. Předpokládejme, že v bázi dat existuje relace PODEZŘELÍ, která má oblasti :číslo zaměstnance, jméno, číslo oddělení, mzda. Vkládání do této relace s výběrem n-tic u existující relace je ukázáno v příkladě 2.

Příklad 2

Přidej do relace PODEZŘELÍ všechny zaměstnance, jejichž prémie jsou větší než polovina jejich mzdy.

```
insert into PODEZŘELÍ:
        select Č-ZAM,JMÉNO,Č-ODDZ,MZDA from ZAM
                where PRÉMIE > 0.5 * MZDA
```

Rušení je proces specifikování n-tic, které mají být odstraněny z báze dat, pomocí doložky where.

Příklad 3

Zruš v relaci ZAM zaměstnance s číslem 857.

```
delete ZAM where Č-ZAM = 857
```

Příkazy pro změnu hodnot oblastí n-tic jsou podobné příkazům pro rušení n-tic. Nové hodnoty pro vybrané oblasti vybraných n-tic dané relace mohou být konstanty nebo výsledky výběru, jak ukazuje příklad 4.

Příklad 4

Změň relaci ZAM zvýšením mzdy o 10% těm zaměstnancům, jejichž zaměstnanecká čísla se vyskytují v relaci PODEZŘELÍ.

```
update ZAM set MZDA =MZDA * 1.1 where
        Č-ZAM in select Č-ZAM from PODEZŘELÍ
```

Je vhodné zde opět připomenout, že SEQUEL nezaručuje jedinečnost výskytu n-tice implicitně, ale pouze na požádání uživatelem pomocí unique.

4.3. Prostředky pro definování dat

Prostředky pro definování dat umožňují uživateli zakládat relace a pohledy na ně. V příkladu 1 je ukázána možná definice relace ODD, jež byla uvedena dříve.

Příklad 1

```
create table ODD (Č-ODD (char (5), nonull),
                  JMÉNO-ODD (char (12),var),
                  MÍSTO (char (20),var))
```

Důležitým prvkem jazyka SEQUEL je možnost definovat pohledy na data. Vlastnosti pohledu jsou specifikovány podobným způsobem jako při dotazu.

Příklad 2

Definuj pohled PROG, který obsahuje jména a mzdy všech programátorů.

```
define view PROG (JMÉNO,MZDA) as
       select ZAM.JMÉNO,ZAM.MZDA from ZAM
                  where ZAM.PRÁCE = 'PROGRAMÁTOR'
```

Rozdíl mezi relací a pohledem na relaci spočívá v tom, že po vytvoření relace se její obsah nemění v závislosti na původní relaci, tj. jsou pamatovány hodnoty, které byly aktuální v okamžiku tvoření nové relace. Při vytvoření pohledu na relaci se pozdější změny v relaci neustále přenášejí i do pohledu, tj. v pohledu pak máme neustále aktuální hodnoty.

Významným rysem jazyka SEQUEL je, že s pohledem na nějakou relaci se pracuje stejně jako s normální relací, tj. umožňuje např. vytvořit pohled na již existující pohledy. Lze tedy říci, že pohled je "dynamické okno" na bázi dat. Nelze však provádět operace insert, update.

Někdy je nutné rozšířit již existující relaci o další oblast. Existující n-tice pak mají v této oblasti hodnotu neurčenou. Dotazy a pohledy, které byly napsány pro původní relaci, zůstanou nezměněny.

Příklad 3

Přidej oblast POČET do relace ODD, která je módu integer.

expand table ODD add column POČET (integer)

Pro zrušení relace nebo pohledu lze použít příkaz drop.

Příklad 4

Zruš pohled PROG.

drop view PROG

Relace nebo pohledy mohou být zrušeny pouze uživatelem, který je založil. Je zřejmé, že při zrušení relace, na kterou je vytvořen pohled, se automaticky zruší i tento pohled.

4.4. Prostředky pro řízení dat

Významnou vlastností jazyka SEQUEL je možnost shlukování příkazů do transakcí tím, že se příkazy vloží mezi

begin transaction a end transaction

To umožňuje automatický návrat stavu, který byl před začátkem transakce v případě, že během transakce se vyskytne chyba. Transakce mohou tvořit strukturu, která je podobná blokové struktuře v Algolu 60.

Pomocí příkazu save lze vytvořit bod opakování, od kterého lze výpočet znovu opakovat pomocí příkazu restore.

Jazyk SEQUEL též umožňuje předdefinovat určitou činnost, která je spojena s relací a operací, která se provádí. To umožňuje snažší udržování integrity báze dat.

Vzhledem k tomu, že jazyk SEQUEL není předmětem předložené práce, byly výše stručně uvedeny nejhlavnější rysy jazyka SEQUEL. Podrobnější informace lze nalézt v [1],[10],[13],[46] BNF jazyka SEQUEL je uvedena v příloze A.

4.5. Příklad na použití jazyka SEQUEL

Způsob použití jazyka SEQUEL ukážeme na jednoduchém příkladě. Předpokládejme, že máme dvě relace:

PLATBY (Č-Z , DATUM , ÚČET)

Č-Z	DATUM	ÚČET
15	78/05/15	4500
15	78/05/06	4700
16	78/07/05	4200
16	78/07/06	4800
17	78/03/02	300

ULOŽENO (Č-Z , DEBIT)

Č-Z	DEBIT
15	12000
16	500
17	8000

,

kde: Č-Z je číslo zákazníka

Naším úkolem je:

1) vytisknout všechny n-tice relace PLATBY
2) vytisknout pro všechny Č-Z v PLATBY placenou sumu
3) změnit relaci ULOŽENO placenou sumou
4) vytisknout změněné n-tice v relaci ULOŽENO
5) vytisknout celkovou placenou sumu

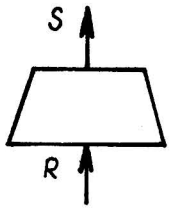
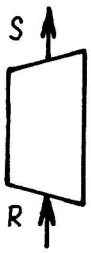
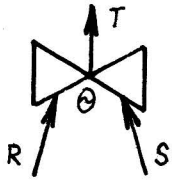
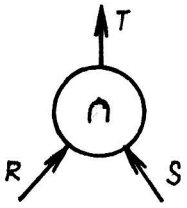
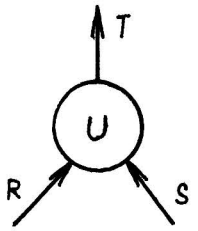
Program na vyrovnání účtu zákazníka je napsán v obecném jazyce s použitím příkazů pro SEQUEL, které jsou označeny obdélníkem. Z příkladu je i vidět, jak lze příkazy SEQUELu začlenit do programu ve vyšším programovacím jazyce.

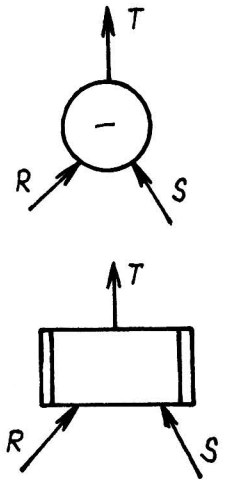
```
open PLATBY ; read PLATBY into PRE ; print PRE ;
SUMTM:=0;  SUMUNM:=0;
while not eof-PLATBY do
      X:=PLATBY.Č-Z;
      CALL BIND('X', ADDR(X)) ;
      CALL BIND('Y', ADDR(Y));
```

```
      CALL SEQUEL (C1,SELECT Č-Z,DEBIT:Y FROM ULOŽENO
                        WHERE ULOŽENO.Č-Z=X);
      CALL FETCH (C1);
      if C1 = empty then SUMG:=0;
         while eof-PLATBY or X = ULOŽENO.Č-Z do
               SUMG+:=ÚČET ;
               SUMTM+:=ÚČET ;
               read PLATBY into PRE;
               print PRE
         od;
         print SUMG;
         Y+:=SUMG;
         CALL BIND ('X', ADDR(X));
         CALL BIND ('X', ADDR(Y));
         CALL SEQUEL (UPDATE ULOŽENO,SET DEBIT = Y
                        WHERE ULOŽENO.Č-Z=X);
         print x,y
      else
         SUMUNM+:=ÚČET ;
         read PLATBY into PRE; print PRE;
         ERROR ('chyba v relacích')
      fi
od;
      SUMTM+:=SUMUNM ;
      print SUMTM;
      close PLATBY;
      CLOSE (C1);
      stop
```

5. Grafické znázornění algebraických operací s relacemi

Pro rychlé pochopení struktury a funkce programu se používají vývojové diagramy, které umožňují programátorovi vyhnout se mnoha chybám při realizaci vlastního algoritmu, zejména pak při různých pozdějších úpravách. Podobně tomu zřejmě bude při používání grafických symbolů pro operace s relacemi.

symbol	operace
	projekce
	výběr
	Θ - spojení
	průnik
	sjednocení



rozdíl

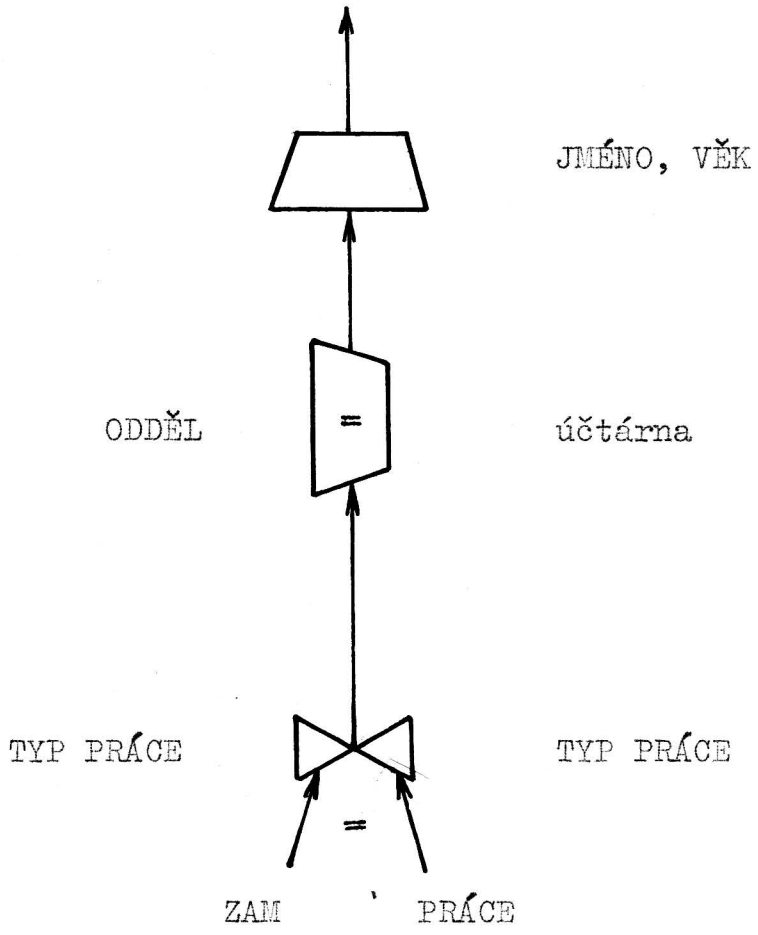
kartézský součin

Výše uvedené grafické symboly jednotlivých operací lze používat pro snažší přehlednost, zejména při výkladu principů optimalizace operací nad relacemi. Abychom osvětlili jejich použití, ukážeme grafické znázornění pomocí jednoduchého příkladu. Uvažujme dvě relace, a to
ZAM = (Č.ZAM,JMÉNO,VĚK,TYP PRÁCE)
a PRÁCE = (TYP PRÁCE,ODDĚL,MÍSTNOST)

Pak například dotaz: nalezni jméno a věk všech zaměstnanců pracujících v oddělení účtárna, může být vyjádřen následující sekvencí příkazů:

R1:= ZAM [TYP PRÁCE = TYP PRÁCE] PRÁCE
R2:= R1 : (ODDEL = 'účtárna')
R3:= R2% (JMÉNO,VĚK)

Tato sekvence příkazů může být znázorněna grafickými symboly takto:



Předložený příklad dokládá, že znázornění pomocí grafických symbolů je přehlednější než vlastní příkazy, i když jde o velmi jednoduchý příklad. U složitějších příkladů tato vlastnost bude vynikat.

6. Návrh vnitřní reprezentace relací

6.1. Úvod

Při návrhu vnitřní reprezentace bylo abstrahováno od omezení týkajícího se velikosti vnitřní paměti. Je tedy předpokládán počítač s virtuální pamětí.

Hlavním požadavkem kladeným na databázový systém je schopnost zodpovědět relativně jednoduchý dotaz v co nejkratší době. Doba, po kterou probíhají operace zakládání a rušení informací v databázovém systému, není již tak kritická. Dosud existující relační databázové systémy jsou založeny na index-sekvenčních souborech např. PRTV [61] a MORIS [4] nebo na speciálních ukládacích technikách např. RM a XRM [2], které vyžadují speciální úpravy v operačních systémech.

V předkládaném návrhu vnitřní reprezentace relací byla použita myšlenka ukládání, známá pod názvem MASTER-DETAIL [20], která byla podstatným způsobem modifikována tak, aby vyhověla požadavkům, které jsou kladeny na databázový systém.

Při ukládání způsobem MASTER-DETAIL se hodnoty položek jednotlivých vět uloží do tabulek. Místo hodnot pak ve větě figurují jen samé směrníky. Uvažme jednoduchý příklad, kdy struktura věty má tvar, který lze v PL/I zapsat takto:

```
DCL 1 ÚDAJE,
        2   JMÉNO   CHAR(10),
        2   OS.ÚDAJE,
              3 NAROZENÍ,
                    4   DEN FIXED (2),
                    4   MĚSÍC FIXED (2),
                    4   ROK FIXED (4),
              3 STAV    CHAR (10);
```

a mějme následující hodnoty pro jednotlivé věty r_i

r_1 := (NOVÁK J; 01; 05; 1961; SVOBODNÝ)

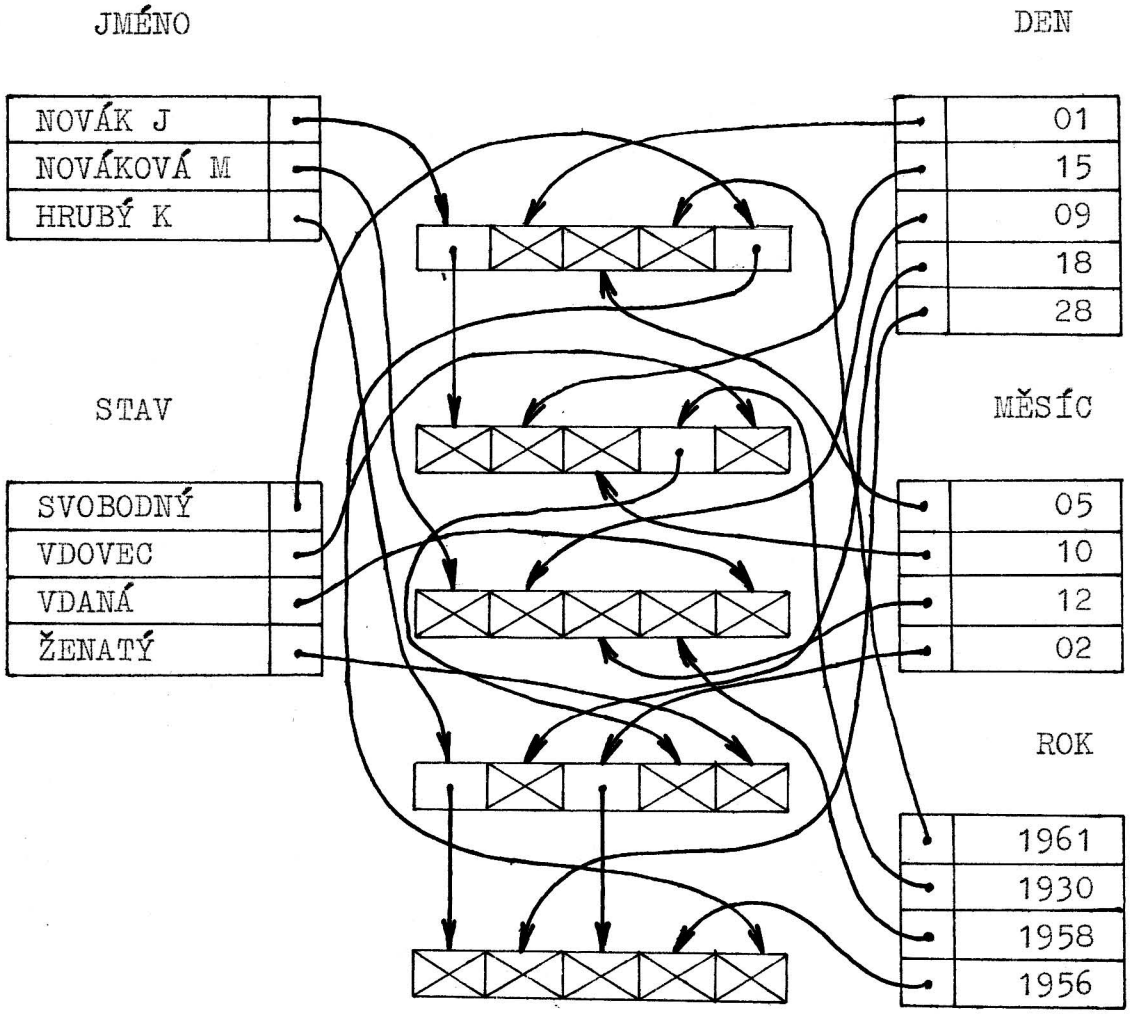
r_2 := (NOVÁK J; 15; 10; 1930; VDOVEC)

r_3 := (NOVÁKOVÁ M; 09; 12; 1958; VDANÁ)

r_4 := (HRUBÝ K; 18; 02; 1930; ŽENATÝ)

r_5 := (HRUBÝ K; 28; 02; 1956; SVOBODNÝ)

Tento soubor při realizaci MASTER-DETAIL má tvar (zpětné směrníky z jednotlivých vět r_i do tabulek jsou pro jednoduchost znázornění vynechány).



obr. 6.1.1.

Všimněme si nyní základních vlastností tohoto způsobu implementace souboru:

Výhody :
a) hodnoty jsou uchovávány pouze jedenkrát,
b) každá položka je klíčová,
c) velmi snadné rozšíření v případě proměnné délky položek.

Nevýhody:
a) nutnost sekvenčního prohledávání tabulek, neboť přidáním věty se nám poruší zvolené uspořádání tabulky,
b) nutnost změny zpětných směrníků ve fyzickém představiteli uzlů v případě přeuspořádání některé z tabulek.

Z výše uvedených vlastností je zřejmé, že po eliminaci nevýhod, je metoda velmi vhodná pro implementaci relací v databázovém systému.

Pokusme se nyní formulovat požadavky kladené na tabulky:

a) možnost nesekvenčního prohledávání tabulek,
b) vyloučení vlivu reorganizace tabulky na ostatní části (tj. ostatní tabulky a uzly),
c) zachování možnosti uchovávat položky proměnné délky v tabulce.

Podrobíme-li organizaci dat zobrazenou na obr.6.1.1. analýze vlastností z hlediska výběru, je zřejmé, že musíme procházet řetězem, abychom zjistili, které věty vyhovují danému kritériu. Procházení řetězem může být časově velmi náročné. Z tohoto důvodu se jeví organizace dat zobrazeňá na obr.6.1.2. výhodnější. Pro úplnost jsou zde zapsány i hodnoty zpětných směrníků do jednotlivých tabulek.

tabulky:

JMÉNO

NOVÁK J	1	2
NOVÁKOVÁ M	3	
HRUBÝ K	5	4

DEN

1	01
2	15
3	09
4	18
5	28

STAV

SVOBODNÝ	1	5
OVDOVĚLÝ	2	
VDANÁ	3	
ŽENATÝ	4	

MĚSÍC

	1	05
	2	10
	3	12
4	5	02

ROK

	1	1961
4	2	1930
	3	1958
	5	1956

uzly:

1	1	1	1	1
1	2	2	2	2
2	3	3	3	3
3	4	4	2	4
3	5	4	4	1

obr.6.1.2.

Provedeme-li např. výběr podle kritéria JMÉNO= ´NOVÁK J´, dostáváme přímo z tabulky JMÉNO množinu $\{1,2\}$ čísel uzlů, pro které je výběrové kritérium splněno bez nutnosti procházet řetězem, jak tomu bylo v případě organizace ukázané na obr.6.1.1. Provedeme-li např. nyní výběr podle kritéria:

$$P = (\text{JMÉNO} = \text{´NOVÁK J.´} \vee \text{JMÉNO} = \text{´HRUBÝ K.´}) \wedge (\text{STAV} = \text{´SVOBODNÝ´} \vee \text{STAV} = \text{´OVDOVĚLÝ ´}),$$

dostáváme množinu množin:

$$\{\{1,2\}, \{5,4\}, \{1,5\}, \{2\}\}$$

přičemž množina čísel uzlů, které splňují kritérium , je výsledkem množinových operací s množinami výše uvedenými a je rovna :

$$(\{1,2\} \cup \{5,4\}) \cap (\{1,5\} \cup \{2\}) = \{1,2,5,4\} \cap \{1,5,2\} = \{1,5,2\}$$

Je zřejmé, že pokud budou seznamy čísel uzlů v tabulkách pro určitou hodnotu uspořádány (např. vzestupně), jde o triviální operace. Také operace negace kritéria je velmi jednoduchá, neboť je převedena na množinový rozdíl. Např. mějme dotaz, jež byl výše uveden a navíc požadujme, aby vybrané osoby nebyly narozeny v roce 1930, tj. máme výběrové kritérium $P \wedge \neg$ ROK = 1930 a po převedení na operace s množinami dostáváme :

$$\{1,5,2\} - \{2,4\} = \{1,5\}$$

Navíc lze jednoduše zajistit jednoznačný výskyt n-tice v relaci.

Z hlediska implementačního jsou tabulky realizovány invertovaným seznamem (INVERTED LIST). Výhodou je rychlost výběru, za kterou však platíme obtížemi při realizaci různě dlouhých seznamů. Tyto obtíže je možno obejít vytvořením seznamů konstantní délky, které se pak řetězí.

6.2. Nefundamentální oblast

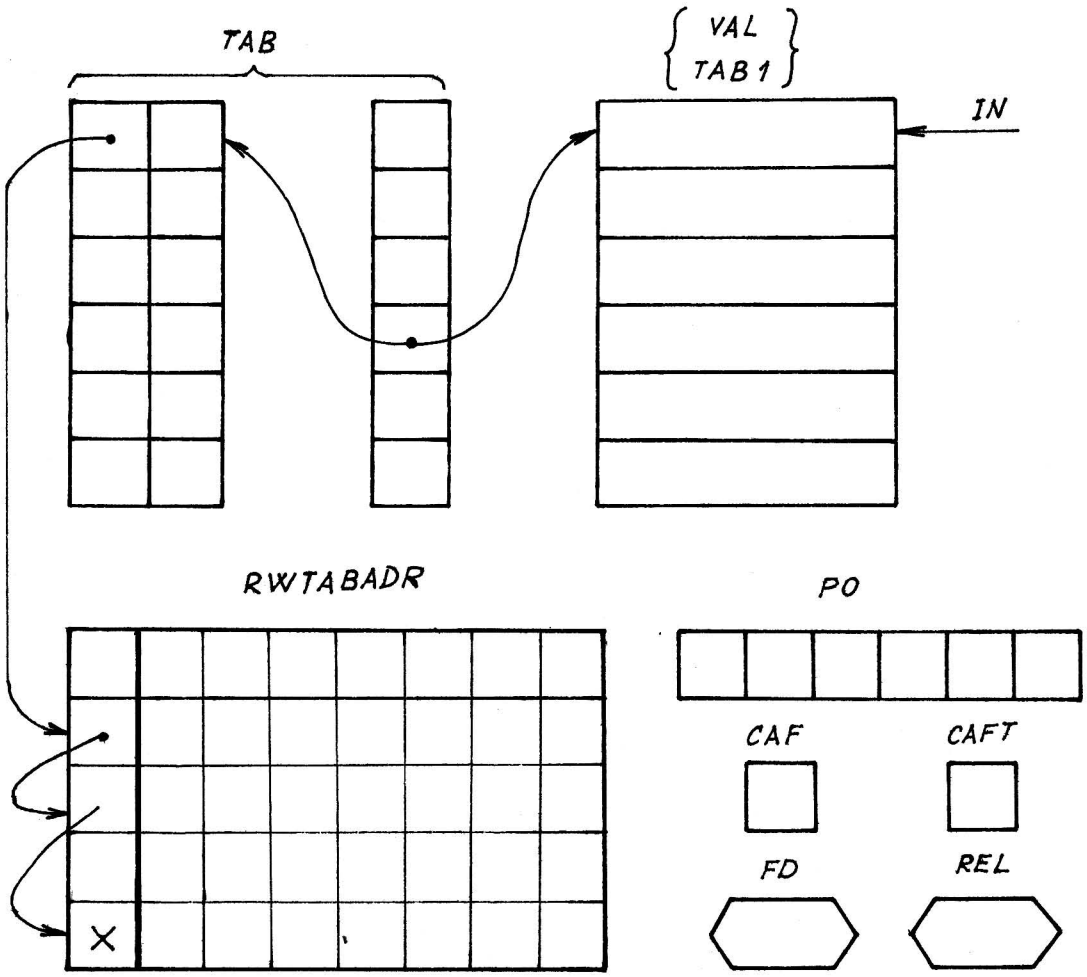
Budeme-li trvat na požadavcích, které jsou kladeny na tabulky a budeme-li navíc požadovat dodržení neustálé uspořádanosti prvků seznamu čísel uzlů, lze navrhnout takovou datovou strukturu, která tyto požadavky splňuje a při vyjádření v Algolu 68 má tvar :

```
mode nfd = struct  (ref [ , ] int TAB,
                    ref [ ] string VAL,
                    ref [ ] int TAB1,
                    ref [ , ] int RWTABADR,
                    [1:6] int PO,
                    int CAF, CAFT,
                    ref fd FD,
                    ref rel REL );
```

obr.6.2.1.

Proměnou módu nfd budeme nazývat nefundamentální oblastí. Po vygenerování paměťového prostoru si lze proměnou módu nfd představit tak, jak je ukázáno na obr.6.2.2.

Z důvodů, jež budou uvedeny později, se generuje paměťový prostor buď pro tabulku TAB1 nebo paměťový prostor pro tabulku VAL. Zatím budeme uvažovat, že byl vygenerován paměťový prostor pro tabulku VAL.



obr.6.2.2.

Pro možnost rychlého výběru by bylo optimální, aby hodnoty, jež se ukládají do tabulky VAL ve znakové podobě, byly uspořádány takovým způsobem, abychom mohli nalézt požadovanou hodnotu co nejrychleji. Vzhledem k tomu, že je nutné uvažovat neustálé změny (přidávání a rušení hodnot), je tento požadavek těžko splnitelný. Jedním ze způsobů, jak zajistit požadavek rychlého vyhledávání hodnot, je použití dichotomistického vyhledávání na setříděné tabulce hodnot. Abychom mohli tento algoritmus použít, je nutné tabulku hodnot rozdělit na tři části,a to na :

a) setříděnou část,
b) nesetříděnou část,
c) neobsazenou část.

V setříděné části lze použít algoritmu dichotomistického vyhledávání a v nesetříděné části pak musíme používat sekvenčního vyhledávání. V případě přidání hodnoty se neobsazená část zkrátí a nesetříděná část se pak prodlouží.

Pro objasnění významu jednotlivých tabulek a proměných uvedeme nyní jejich funkční význam.

RWTABADR - je zřetězený seznam seznamů, kde jsou uloženy čísla uzlů mající stejnou hodnotu odpovídající oblasti relace. Je to realizace adresové části tabulky např. JMÉNO z obr. 6.1.3. První sloupec v tabulce slouží ke zřetězování.

VAL - tabulka, kde jsou uloženy vlastní hodnoty ve znakové podobě. Do této tabulky ukazují zpětné směrníky, které jsou uloženy v uzlech viz.obr.6.1.2. Na obr. 6.2.2. je zpětný směrník označen IN. Řádky v tabulce VAL proto nelze přeuspořádat bez dodatečné opravy hodnot zpětných směrníků, které jsou uloženy v uzlech.

TAB - na obr.6.2.2. je tabulka "roztržena" na 2 části. V prvním sloupci je uložena adresa na začátek řetězu seznamů čísel uzlů, v druhém sloupci je uložen skutečný počet prvků seznamu a třetí sloupec slouží k přeindexování, čímž je umožněna snadná reorganizace tabulek.

PO - pole, kde jsou uchovány informace o délce setříděné a nesetříděné části, o počtu zrušených položek a o rozměrech jednotlivých tabulek.

CAF,CAFT - slouží k hospodaření s pamětí přidělenou tabulce RWTABADR.

REL - reference ukazující na proměnou, která reprezentuje uzly z obr.6.1.2.

FD - reference ukazující na slovník, kam mohou být hodnoty ukládány, tj. na fundamentální oblast.

TAB1 - tabulka, kam se ukládají čísla řádek ve fundamentální oblasti, kde jsou hodnoty uloženy.

Poznámka 1 :

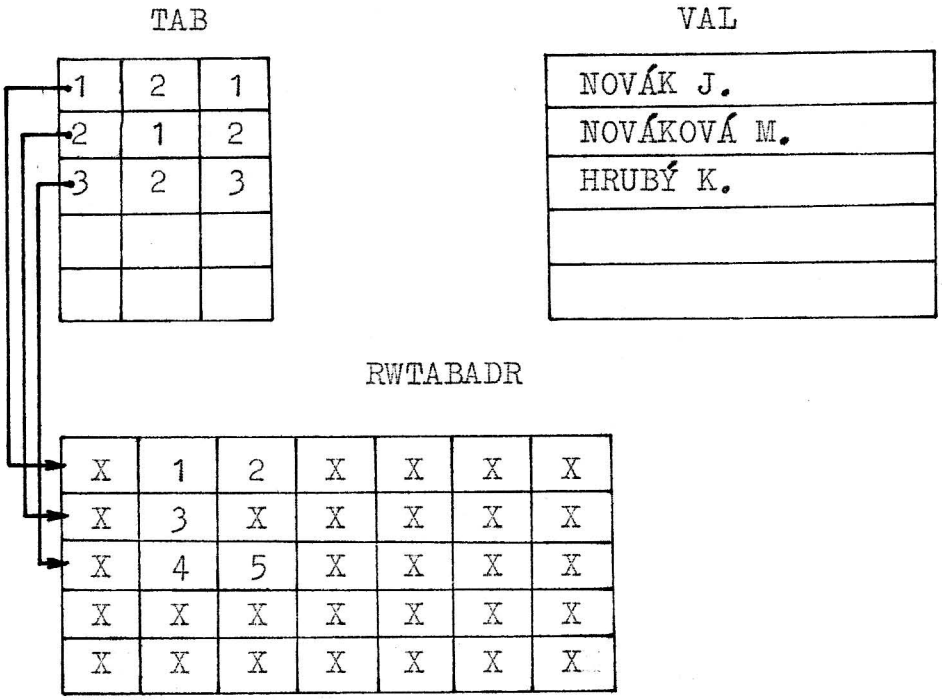
Fundamentální oblast bude vysvětlena později a pak bude jasný i smysl proměnných FD, TAB1.

Poznámka 2:

Pokud hodnoty v oblasti relace mají vlastnost primárního klíče, tj. každá hodnota se v oblasti relace vyskytuje pouze jedenkrát, nemusí se generovat paměťový prostor pro tabulku RWTABADR a hodnoty v prvním sloupci pak ukazují přímo do oblasti reprezentující uzly.

Pro osvětlení ukážeme na obr.6.2.4.nefundamentální oblast po uložení tabulky JMÉNO z obr.6.1.2.

JMÉNO



obr.6.2.4.

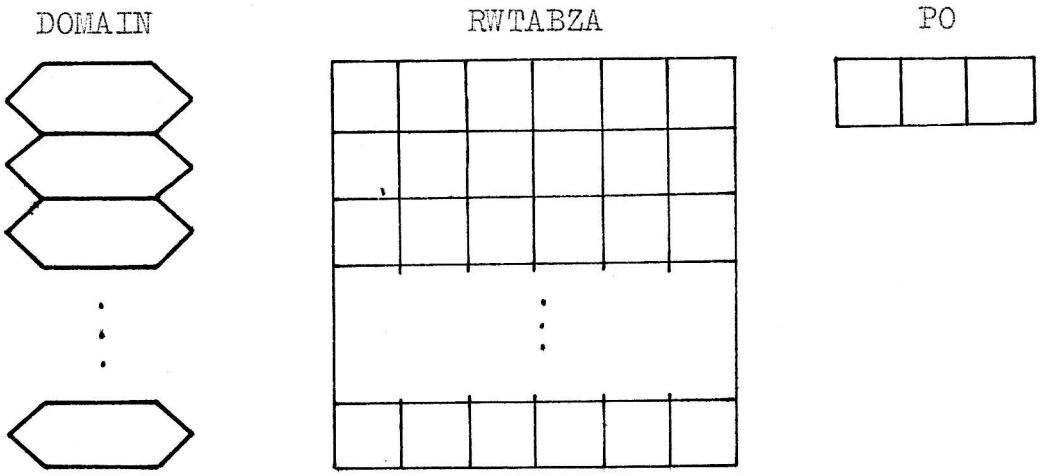
6.3. Oblast reprezentující uzly

V předchozí části jsme se zabývali vlastně jen otázkou, jak reprezentovat oblast relace pomocí datové struktury, kterou jsme nazvali nefundamentální oblastí. Nyní se však musíme zabývat otázkou, jak reprezentovat vazbu jednotlivých hodnot oblastí tak, aby tvořily n-tici relace. Jde tedy o to, jak reprezentovat uzly z obr. 6.1.2. Při tvorbě datové struktury nesmíme však zapomenout, že je nutné "svázat" nějakým způsobem nefundamentální oblasti s oblastí reprezentující uzly, neboť dohromady vytváří jeden logický celek, a to realizaci relace. Datovou strukturu, která toto splňuje, je možno v Algolu 68 popsat takto:

```
mode rel = struct (ref [ ] ref nfd DOMAIN,
                   ref [,] int RWTABZA,
                   [1:3]  int PO ) ;
```

obr.6.3.1.

Po vygenerování paměťového prostoru si proměnou módu rel lze představit tak, jak je ukázáno na obr. 6.3.2.



obr.6.3.1.

Význam jednotlivých proměných :

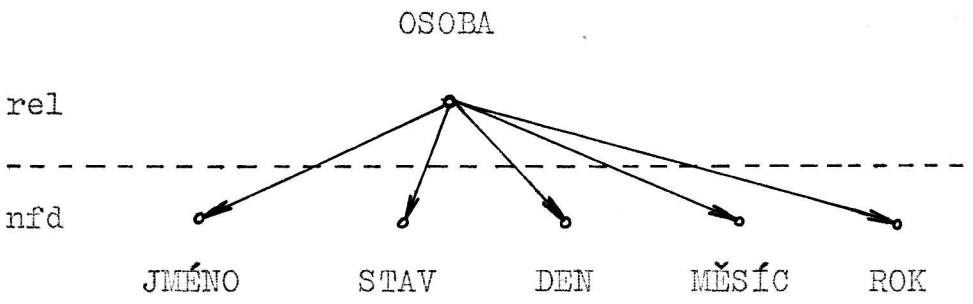
RWTABZA - tabulka zpětných směrníků ukazujících na řádku v tabulce VAL nebo v TAB1 odpovídající nefundamentální oblasti.

DOMAIN - reference na příslušné nefundamentální oblasti dané relace. Pořadí referencí je důležité.

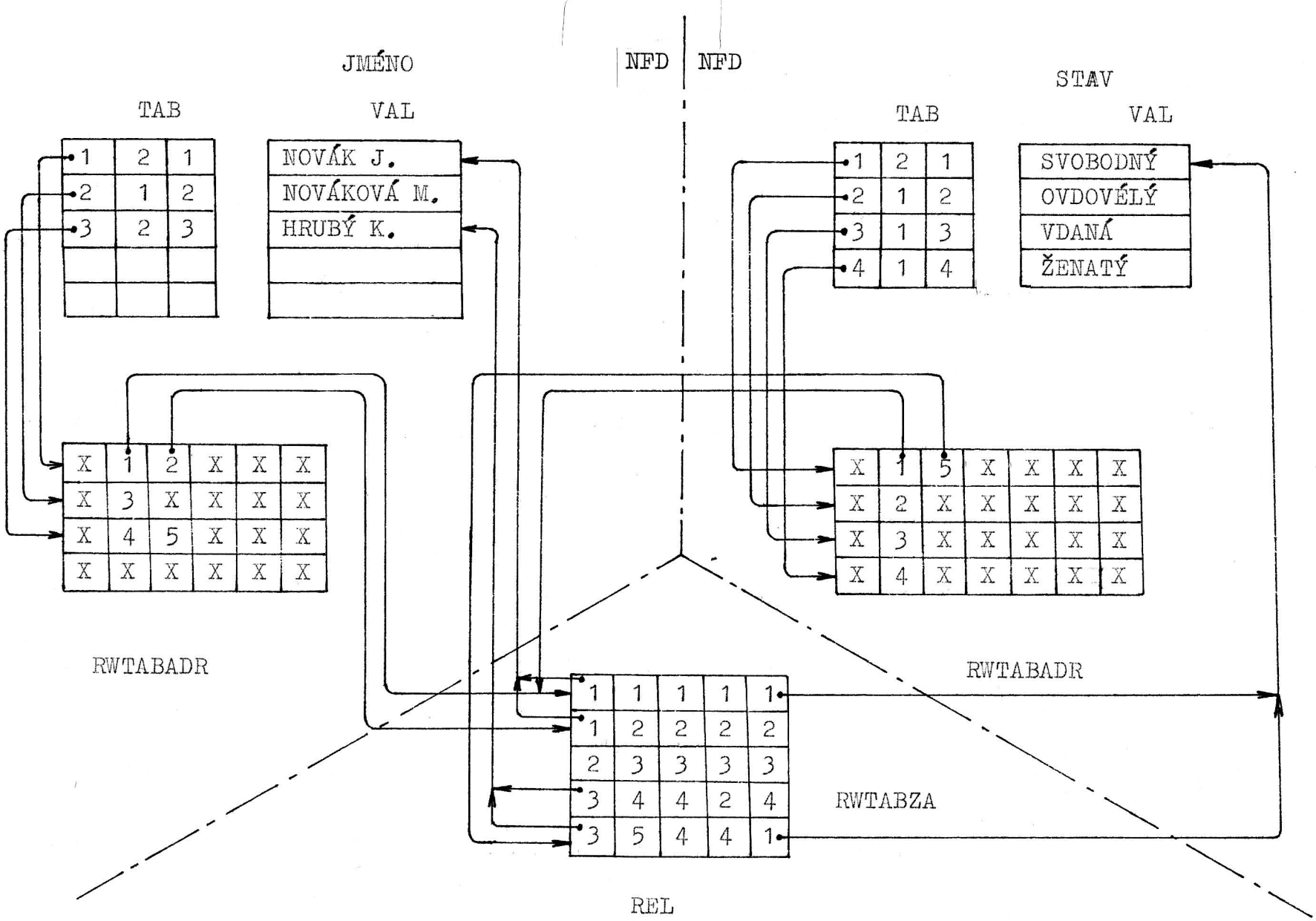
PO - informace o délce neobsazené části RWTABZA, o počtu zrušených řádek v RWTABZA.

Na obr.6.3.4. ukážeme spojení oblasti reprezentující uzly a dvou nefundamentálních oblastí jako celku pro případ, který byl ukázán na obr.6.1.2. Předpokládáme, že dosud neproběhla reorganizace žádné nefundamentální oblasti a ani oblasti reprezentující uzly.

Na obr.6.3.4. jsou znázorněny jen vazby pomocí směrníků jen pro nefundamentální oblast JMÉNO a STAV. Situaci na obr.6.1.2. si lze znázornit takto:



obr.6.3.3.


JMÉNO
NFD
NFD
STAV
TAB
VAL
TAB
VAL
NOVÁK J.
NOVÁKOVÁ M.
HRUBÝ K.
SVOBODNÝ
OVDOVĚLÝ
VDANÁ
ŽENATÝ
RWTABADR
RWTABADR
RWTABZA
REL

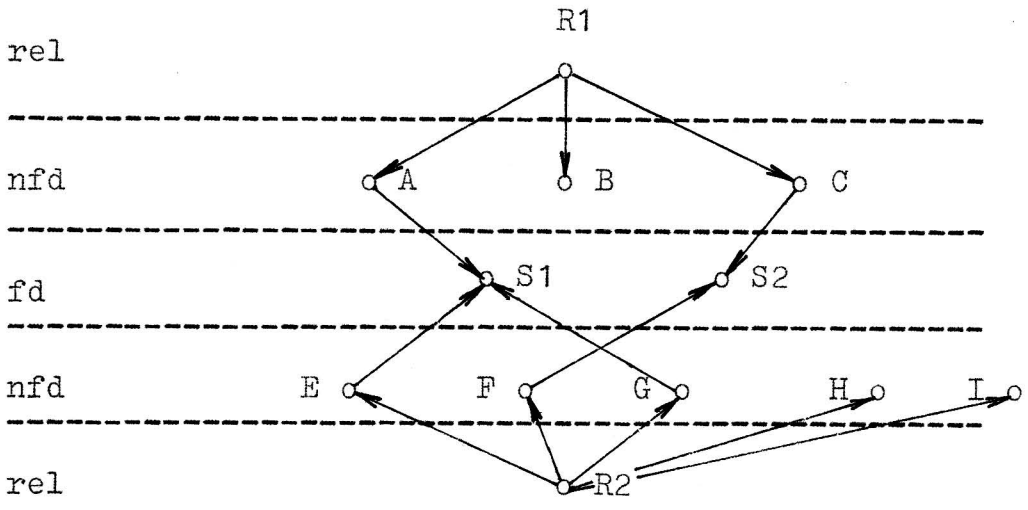
obr.6.3.4.

6.4. Fundamentální oblast

Často může nastat případ, že několik oblastí v relaci má přibližně stejnou množinu hodnot. Pak by v relaci existovalo několik nefundamentálních oblastí, které by měly "skoro" shodné výřezy VAL of NFD. Proto by bylo vhodné, aby takové oblasti měly společný slovník a v příslušných nefundamentálních oblastech by pak bylo generováno místo pro tabulku TAB1, kam by se ukládaly směrníky ukazující do fundamentální oblasti a paměťové místo pro tabulku VAL by pak generováno nebylo. Při vhodné volbě datové struktury by pak mohlo mít společný slovník i několik nefundamentálních oblastí různých relací. Takový slovník budeme dále nazývat fundamentální oblastí. Za těchto podmínek lze realizovat následující případ:

Mějme dvě relace R1 (A,B,C), R2 (E,F,G,H,I)
a slovníky : S1, který je společný pro oblasti R1.A,
R2.E,R2.G,
S2, který je společný pro oblasti R1.C,
R2.F.

Pak si lze příslušnost jednotlivých nefundamentálních oblastí k oblastem reprezentující uzly a k fundamentálním oblastem znázornit takto:



obr.6.4.1.

Nyní je vidět, že položka FD v proměné módu nfd slouží ke"svázání" fundamentální a nefundamentální oblasti. Požadavky, které jsou kladeny na fundamentální oblast, jsou následující :

a) vyloučení vlivu reorganizace fundamentální oblasti na ostatní části, které jsou s ní svázány,

b) možnost nesekvenčního prohledávání tabulek,

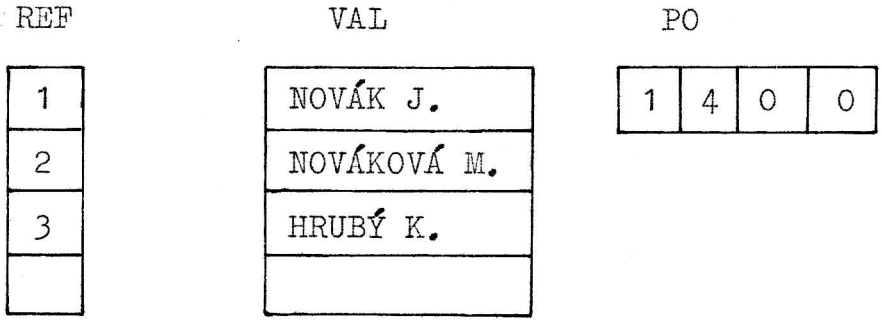
c) možnost uchovávat položky proměnné délky.

Datovou strukturu, která umožňuje výše uvedené požadavky zajistit, je možno v Algolu 68 popsat takto:

```
mode fd = struct (ref [  ] int REF,
                  ref [  ] string VAL,
                  [1 : 4]  int PO);
```

obr.6.4.2.

Po vygenerování paměťového prostoru si lze fundamentální oblast znázornit takto:



obr.6.4.3.

Pro názornost byly do fundamentální oblasti uloženy hodnoty oblasti JMÉNO relace z obr.6.1.2.

Význam jednotlivých tabulek:

VAL - slouží k uchování hodnot ve znakové podobě. Do této tabulky ukazují směrníky z odpovídajících nefundamentálních oblastí.

REF - slouží k tomu, abychom mohli fundamentální oblast přeuspořádat.

PO - informace o délce setříděné části fundamentální oblasti.

Podobně jako u nefundamentální oblasti byla fundamentální oblast rozdělena na setříděnou, nesetříděnou a neobsazenou část. Setříděná i nesetříděná část se prohledává stejným způsobem jako v případě nefundamentální oblasti.

6.5. Operace nad jednotlivými oblastmi

Při návrhu dříve uvedených datových struktur byl stále dodržován požadavek, aby vliv změny vnitřní struktury jedné oblasti neměl vliv na strukturu ostatních oblastí. Dodržení tohoto požadavku např. umožní, aby dvě oblasti, které zastávají stejnou funkci, měly naprosto odlišné datové struktury. Je zřejmé, že algoritmy realizující jednotlivé operace nad oblastmi budou závislé na typu zvolených datových struktur. Proto budou uvedeny algoritmy pro realizaci operací nad jednotlivými oblastmi pouze pro dříve uvedené datové struktury.

6.5.1. Operace nad fundamentální oblastí

Uvedeme nyní jména podprogramů realizujících operace nad fundamentální oblastí a jejich hrubé vývojové diagramy. Podrobné vývojové diagramy jsou uvedeny v příloze B.

INIT - vygenerování potřebného místa v paměti pro umístění hodnot fundamentální oblasti. Dále se zajišťuje prvotní nastavení tabulek REF a VAL.

INSERT - uložení hodnoty ve znakové podobě do fundamentální oblasti. V případě, že hodnota již byla dříve založena do fundamentální oblasti, duplicitní založení se již neprovede a proměná BOOL pak nabývá hodnoty true. V případě, že hodnota byla do fundamentální oblasti fyzicky založena, proměná BOOL nabude hodnoty false. Další výstupní hodnotou je číslo řádky tabulky VAL, kde je hodnota uložena.

GET - pro dané číslo řádky tabulky VAL je vrácena hodnota uložená na tomto řádku.

FIND - pro danou hodnotu se zjistí, zda je uložena v tabulce VAL. Je-li hodnota v této tabulce nalezena, pak proměná BOOL nabývá hodnoty true a v proměné IN je pak uloženo číslo řádky tabulky VAL, kde je hodnota uložena. V případě, že hodnota není nalezena, nabývá proměná BOOL hodnoty false.

DELETE - zruší hodnotu uloženou ve fundamentální oblasti a zvýší hodnotu proměné PO[3] , která udává počet zrušených hodnot o jednu.

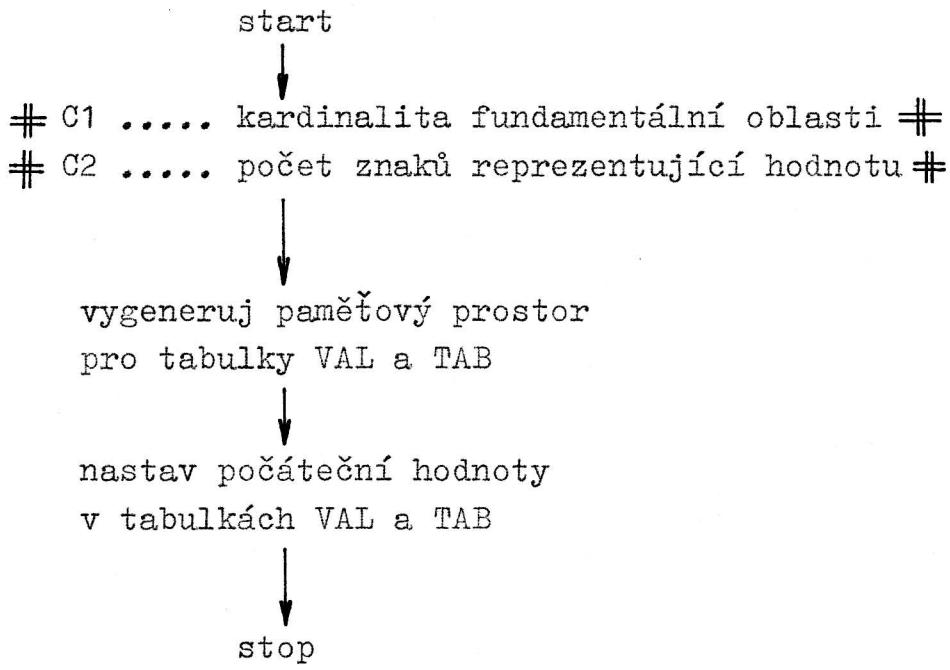
SORT - zajistí přeuspořádání fundamentální oblasti tak, aby bylo možné používat dichotomistického vyhledávání hodnot uložených v tabulce VAL fundamentální oblasti.

ADDSPACE - zajistí zvětšení paměťového prostoru, který je obsazen tabulkami REF a VAL, v případě, že paměťový prostor je plně obsazen hodnotami.

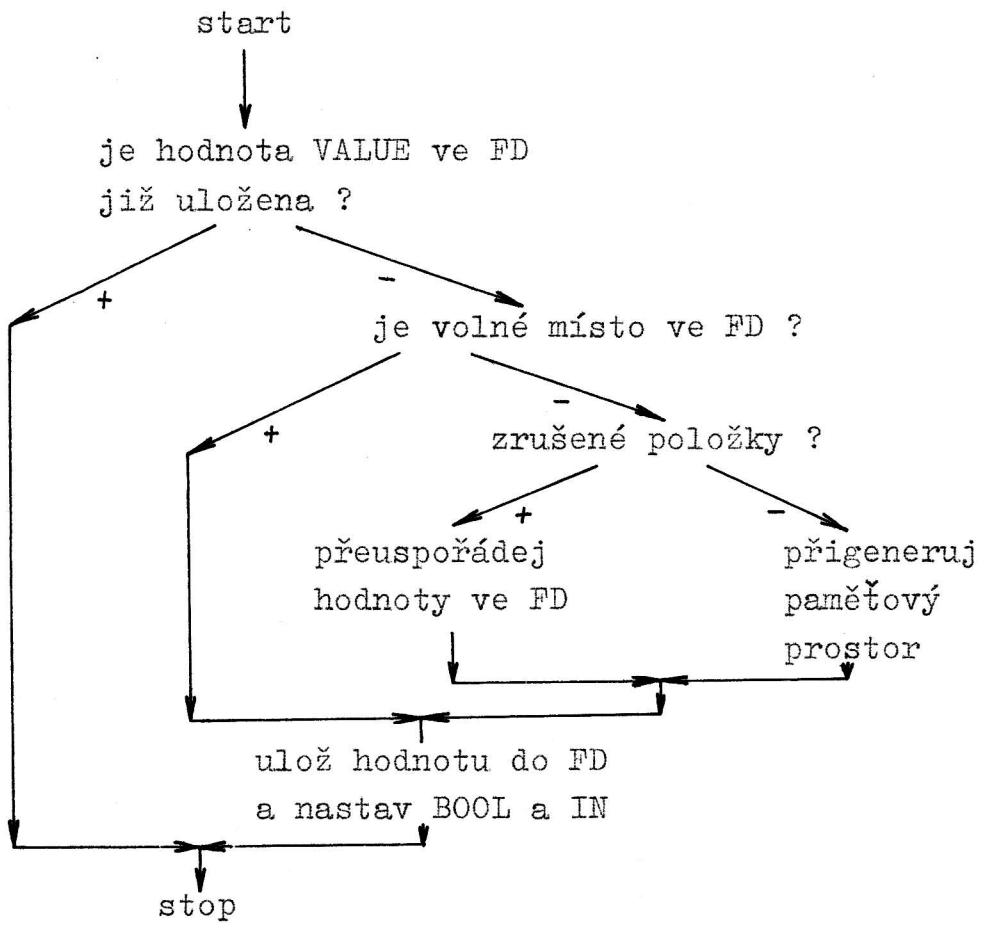
TISKFD - zajistí vytištění hodnot, které jsou uloženy ve fundamentální oblasti. Podprogram slouží pouze pro účely ladění.

ZKUSFD - provede pro danou fundamentální oblast testy na výše uvedené algoritmy. Z testů lze zjistit, zda algoritmy jsou správně realizovány. Podprogram slouží pouze pro účely ladění výše uvedených algoritmů.

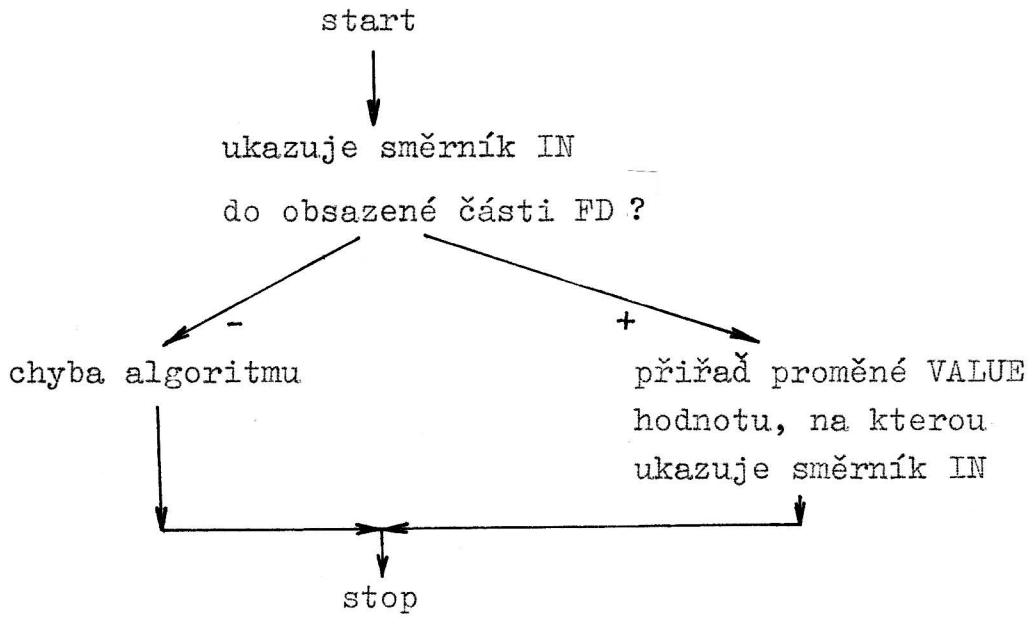
INIT (ref fd FD, int C1,C2) void :



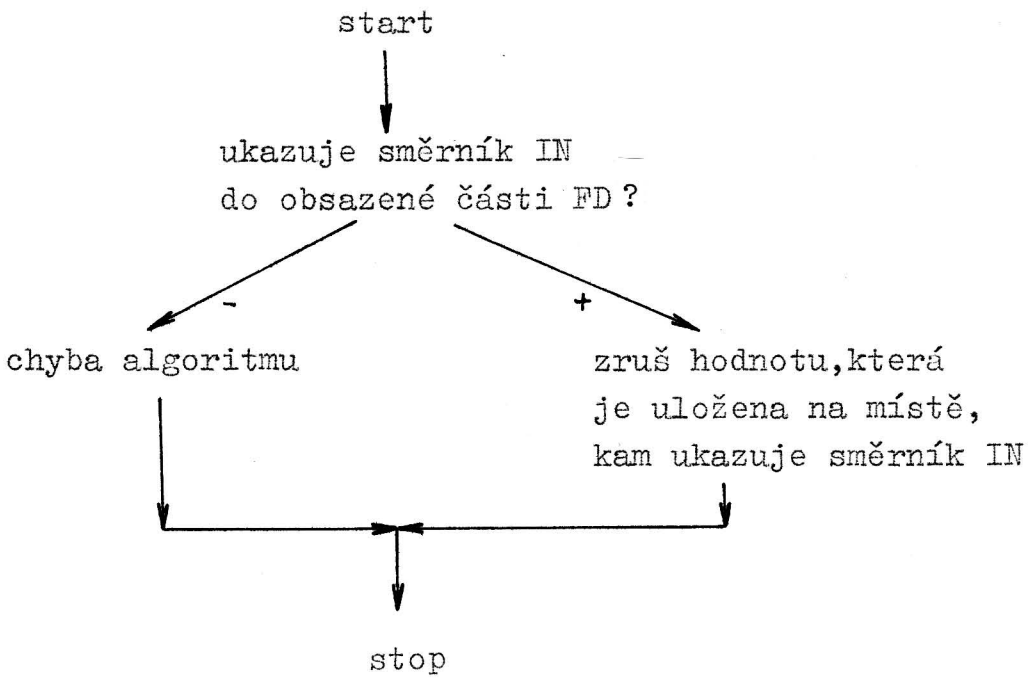
INSERT (ref fd FD,[]char VALUE,ref bool BOOL,ref int IN) void:



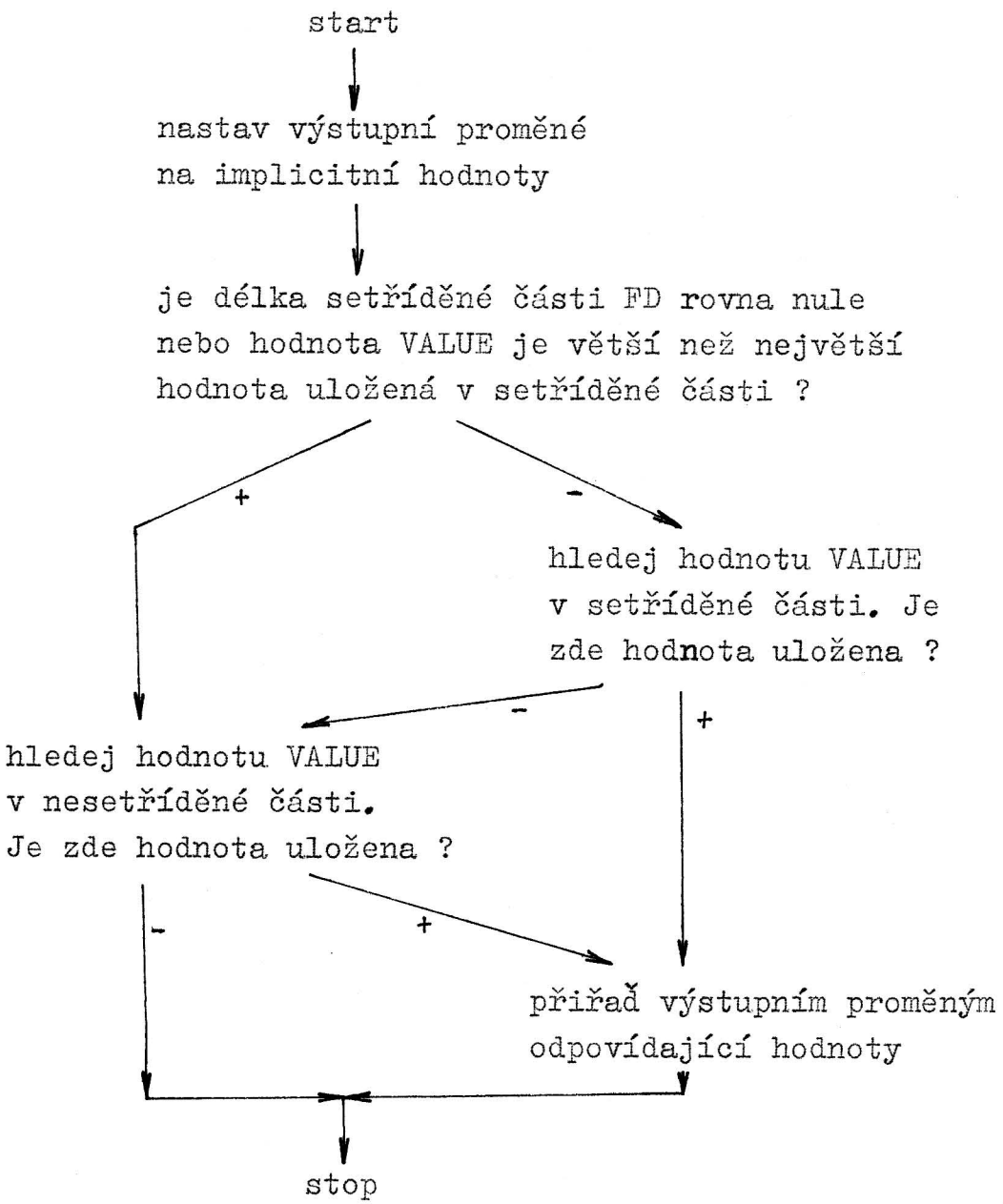
GET (ref fd FD,ref[ ]char VALUE, int IN) void :



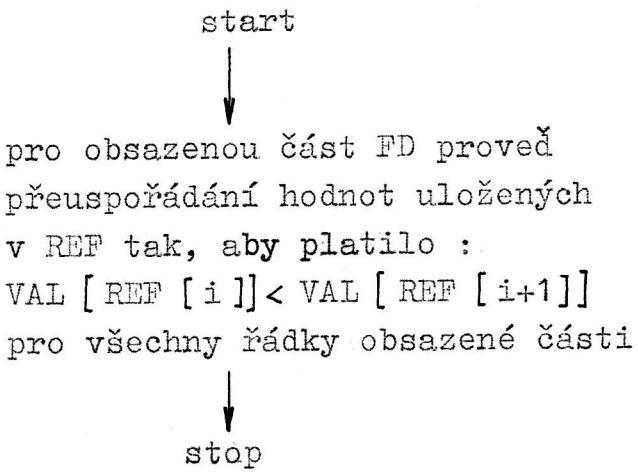
DELETE (ref fd FD, int IN) void:



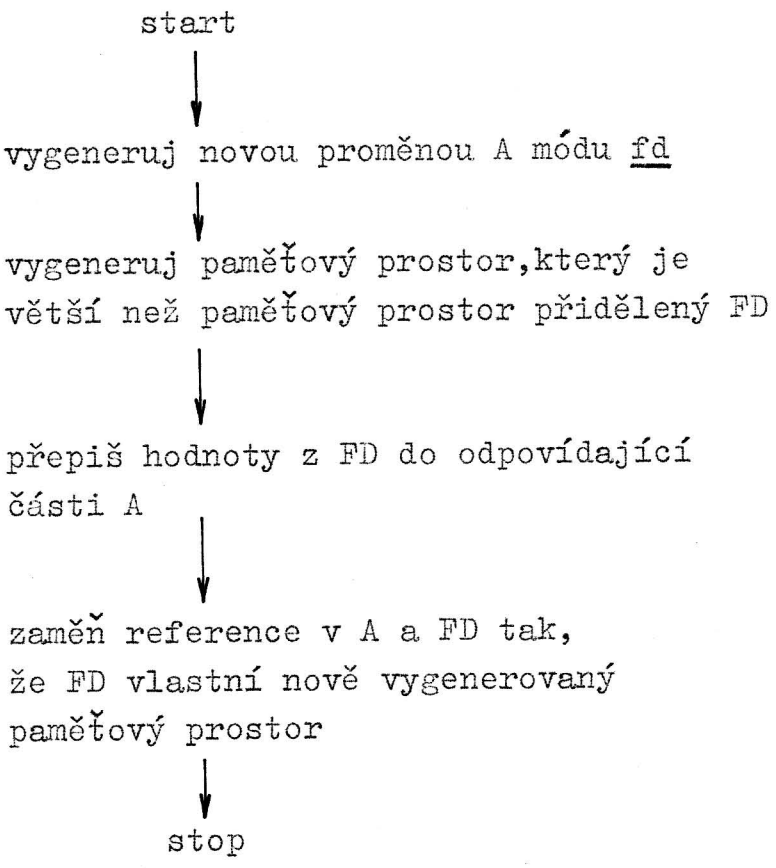
FIND (ref fd FD, [ ] char VALUE, ref bool BOOL, ref int IN)void:



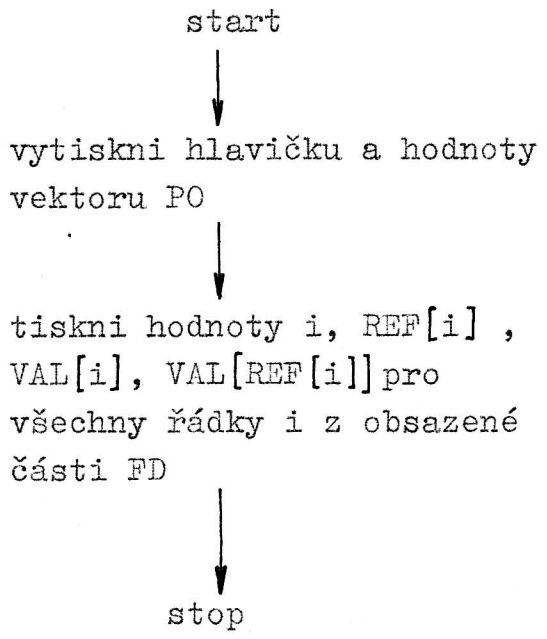
SORT (ref fd FD) void :



ADDSPACE (ref fd FD) void:



TISKFD (ref fd FD) void :



6.5.2. Operace nad nefundamentální oblastí

Nyní uvedeme jména podprogramů realizujících operace nad nefundamentální oblastí a jejich hrubé vývojové diagramy. Podrobné vývojové diagramy jsou uvedeny v příloze C.

INITNFD - vygenerování potřebného místa v paměti pro uložení hodnot nefundamentální oblasti. Prvotní nastavení hodnot tabulek a proměných.

INSERTNFD - uložení hodnoty ve znakové podobě do nefundamentální oblasti v případě, že není definována fundamentální oblast. V opačném případě se hodnota uloží do příslušné fundamentální oblasti a do nefundamentální oblasti se uloží jen hodnota směrníku ukazující do fundamentální oblasti. V obou případech je zajištěna jednoznačnost výskytu hodnoty. Výstupními hodnotami jsou hodnoty proměné BOOL, která indikuje, zda hodnota byla skutečně fyzicky ukládána, či zda byla uložena již dříve, a hodnota IN, která udává číslo řádky tabulky VAL respektive TAB1, kde je uložena hodnota, resp. směrník ukazující na hodnotu v odpovídající fundamentální oblasti.

FINDNFD - pro danou hodnotu se zjistí, zda hodnota je uložena v nefundamentální oblasti. Je-li hodnota uložena, pak výstupní proměná BOOL nabývá hodnoty true a proměná IN je přiřazeno číslo řádku, kde je uložena hodnota, resp. směrník do příslušné fundamentální oblasti.

SORTNFD - zajistí přeuspořádání nefundamentální oblasti tak, aby bylo možné používat dichotomistického vyhledávání hodnot uložených v nefundamentální oblasti.

ADDSPACENFD - zajistí zvětšení paměťového prostoru, který je obsazen tabulkami TAB,RWTABADR a VAL, resp. TAB1 v případě, že paměťový prostor je plně obsazen.

TISKNFDALL - zajistí vytištění všech hodnot, které jsou uloženy v nefundamentální oblasti. Vytisknou se zároveň i hodnoty uložené v příslušné nefundamentální oblasti, pokud existuje. Podprogram slouží pouze pro účely ladění ostatních podprogramů zabezpečujících správnou funkci nefundamentální oblasti.

GETALL - zajistí přepsání hodnot směrníků uložených v tabulce RWTBADR jako zřetězený seznam seznamů do vektoru.

GETALL1 - funkce podobná funkci podprogramu GETALL, ale navíc jsou eliminovány zrušené hodnoty směrníků.

GETNFD - pro dané číslo řádky tabulky VAL, resp.TAB1 je vrácena hodnota uložená na tomto řádku, resp. hodnota uložená v odpovídající fundamentální oblasti, do níž ukazuje směrník uložený v tabulce TAB1.

RETURN - zajistí vrácení uvolněných seznamů v tabulce RWTABADR systému. Tím je umožněno hospodárné využití přidělené paměti.

ADRINTO - zajistí přepsání směrníků uložených ve vektoru do zřetězeného seznamu v tabulce RWTABADR.

FINDNFDSTRING - zajistí vyhledání hodnoty ve znakové podobě v tabulce VAL. Podprogram je využíván podprogramem FINDNFD.

FINDNFDINT - zajistí vyhledání hodnoty směrníku v nefundamentální oblasti v tabulce TAB1. Podprogram je využíván podprogramem FINDNFD.

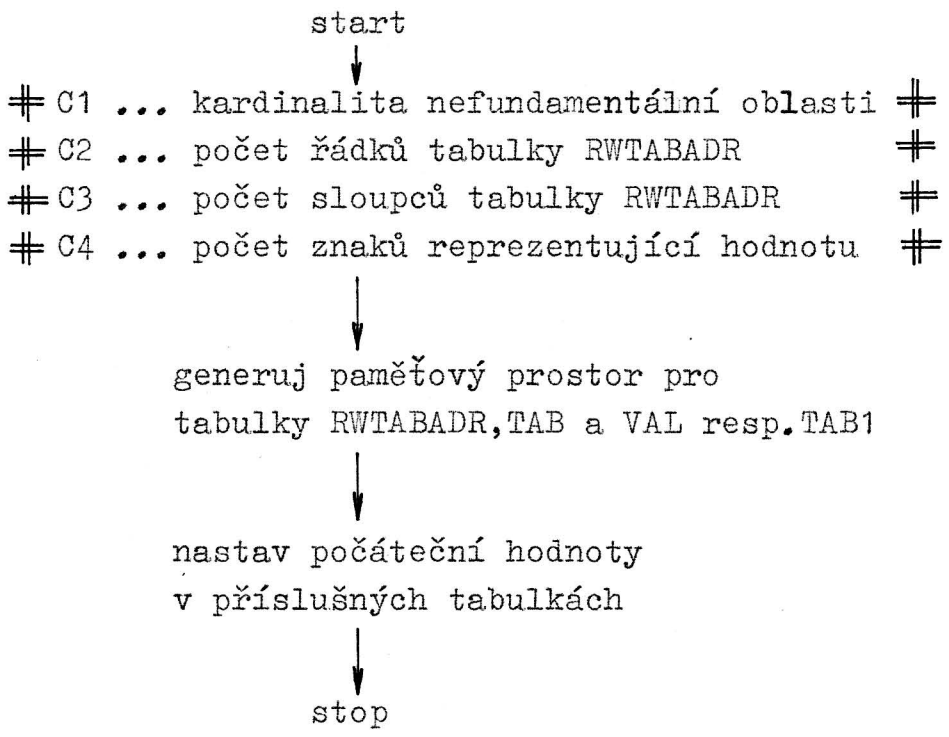
DELETENFD - zajistí zrušení hodnoty uložené v nefundamentální oblasti a zvýší hodnotu proměné udávající počet zrušených položek.

DELETENFDADR - zajistí zrušení směrníku, který je uložen v tabulce RWTABADR.

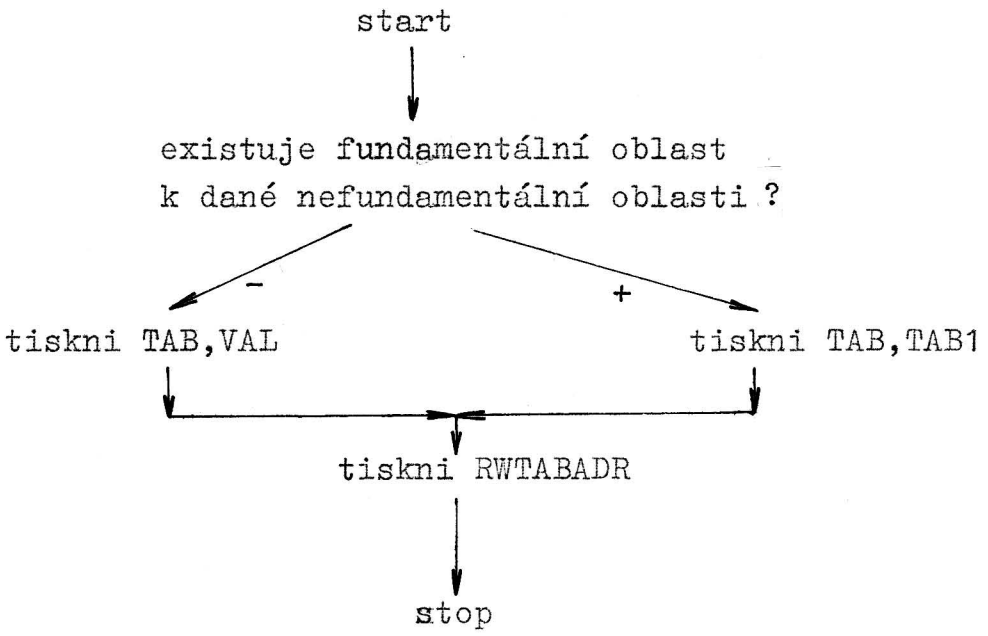
PREINDEXUJRWTABADR - zajistí přeindexování směrníků uložených v tabulce RWTABADR. Přeindexování je někdy vyžadováno při reorganizaci oblasti reprezentující relace.

TISKNFD - zajistí vytištění hodnot uložených pouze v nefundamentální oblasti. Hodnoty, jež jsou uloženy v odpovídající fundamentální oblasti, pokud existuje, se netisknou.

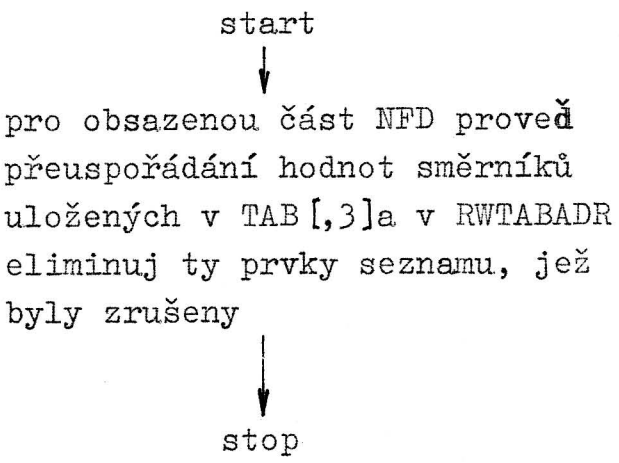
INITNFD (ref nfd NFD, int C1,C2,C3,C4,ref fd FD) void:



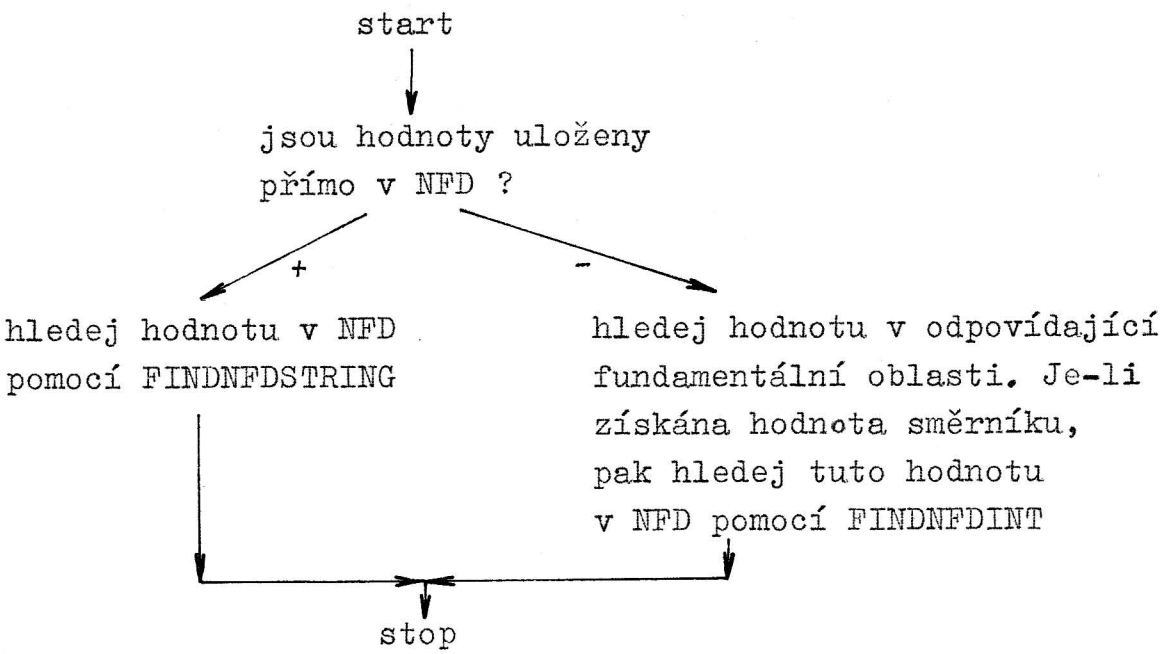
TISKNFD (ref nfd NFD) void :



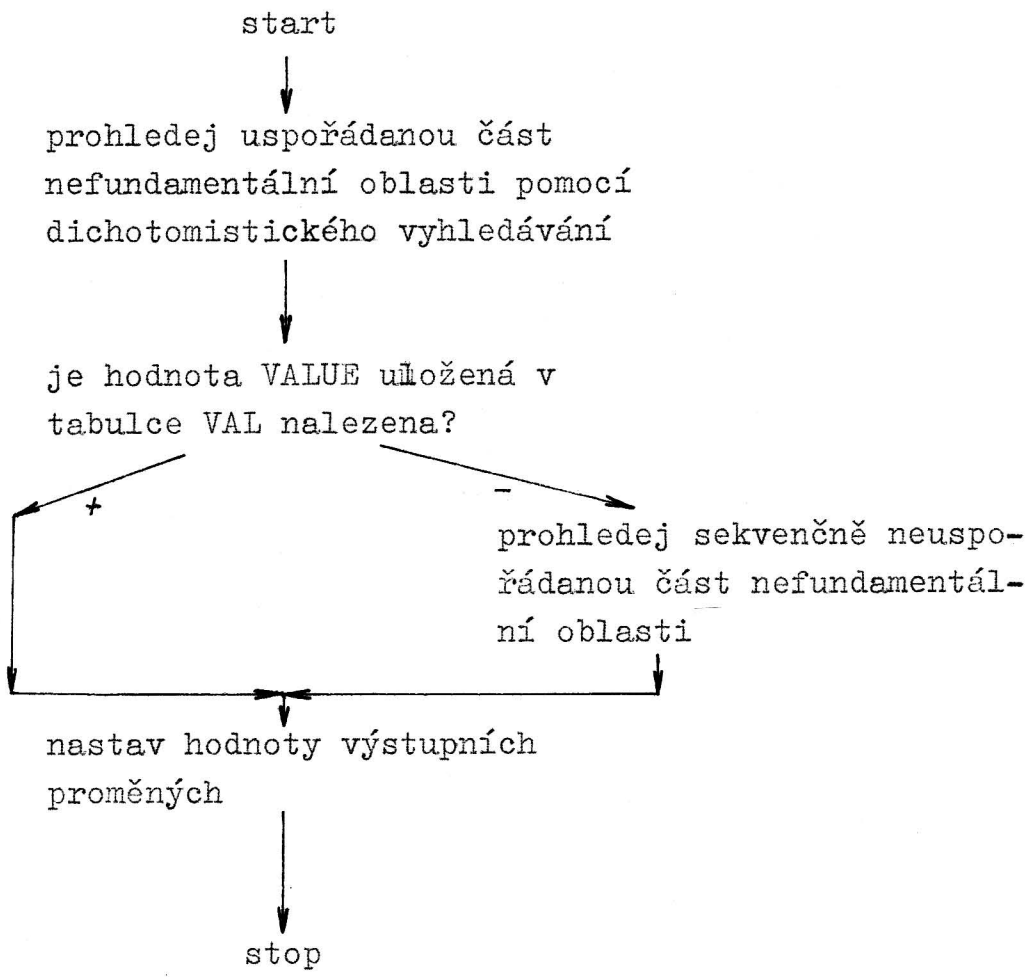
SORTNFD (ref nfd NFD) void:



FINDNFD (ref nfd NFD,[]char VAL,ref int I, ref bool BOOL)void:

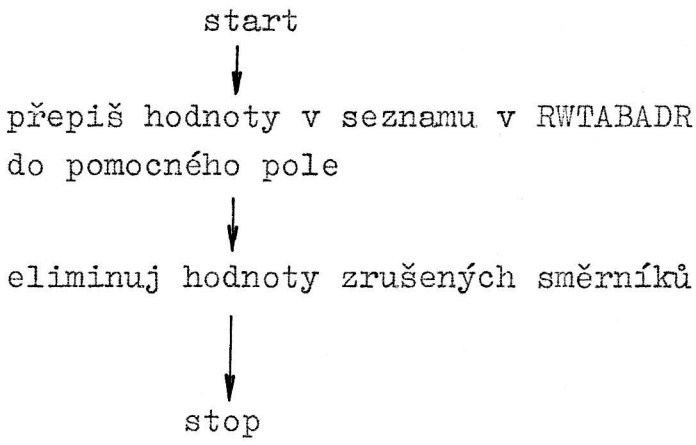


FINDNFDSTRING (ref nfd NFD,[]char VALUE,ref int I,ref bool BOOL)
void:

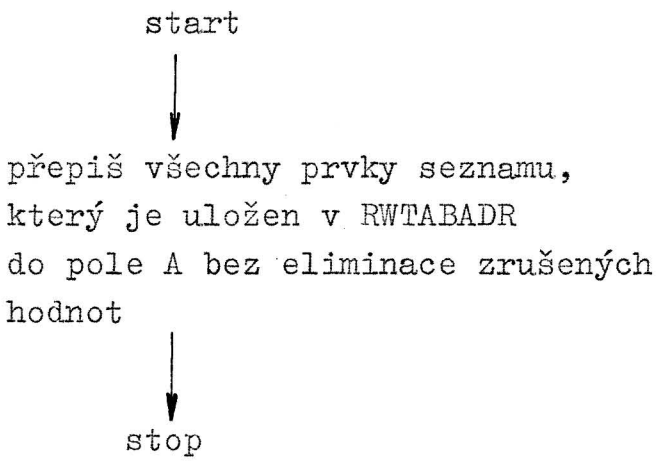


FINDNFDINT (ref nfd NFD,int VALUE,ref int I,ref bool BOOL)
void:
stejná jako FINDNFDSTRING, ale s tím rozdílem, že se hodnoty hledají v tabulce TAB1.

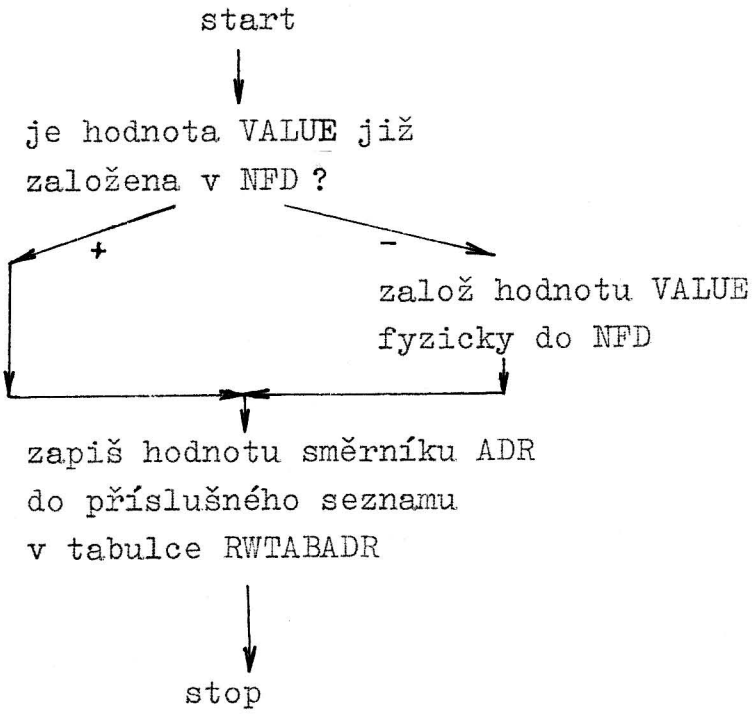
GETALL1 (ref int A,ref [,] int RWTABADR,int ADR,ref int L)
void:



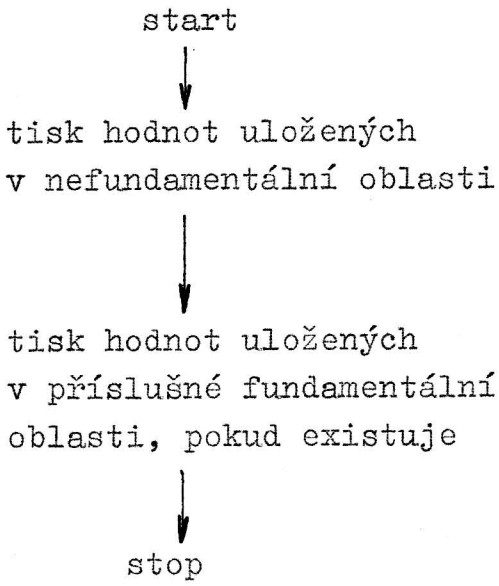
GETALL (ref [ ] int A, ref [,] int RWTABADR,int ADR) void:



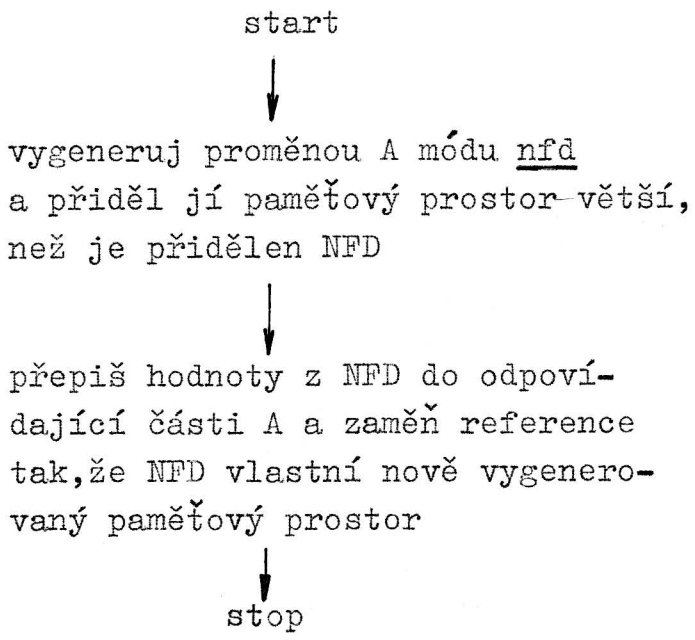
INSERTNFD (ref nfd NFD,int ADR,[ ] char VALUE,ref int INADR)
void:



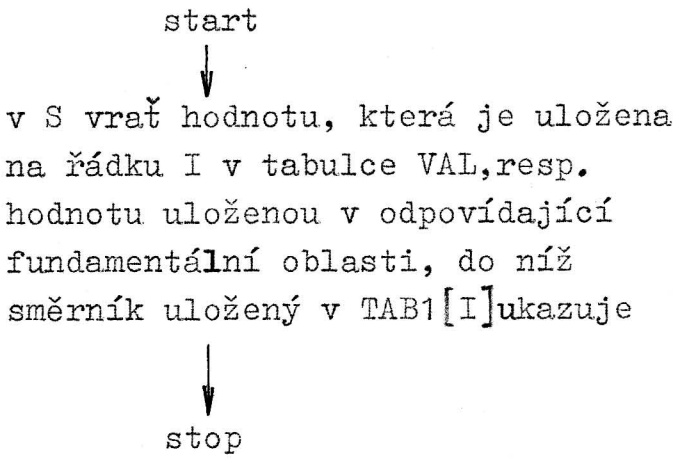
TISKNFDALL (ref nfd NFD) void:



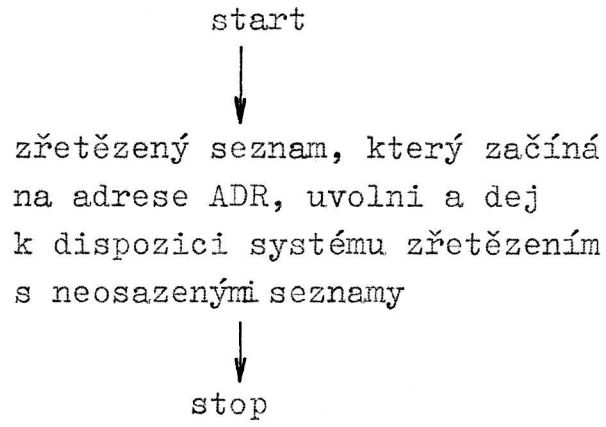
ADDSPACENDF (ref nfd NFD) void:



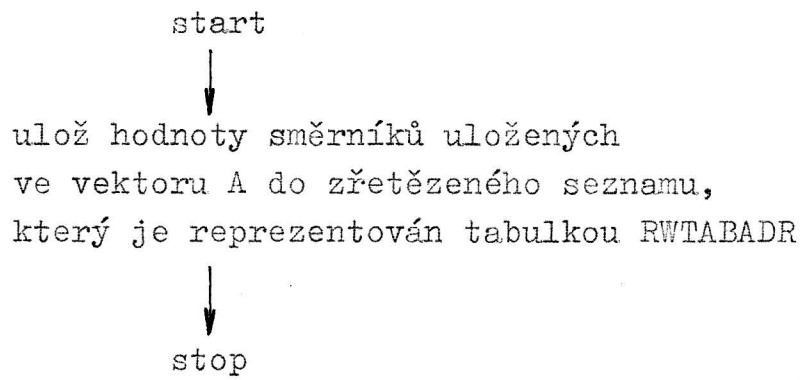
GETNFD (ref nfd NFD,ref[ ]char S, int I) void:



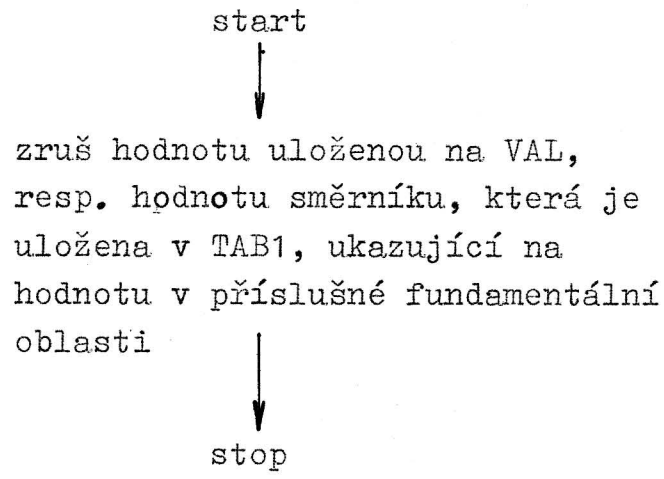
RETURN (ref int ADR,ref [,] int RWTABADR,ref int CAFT)void:



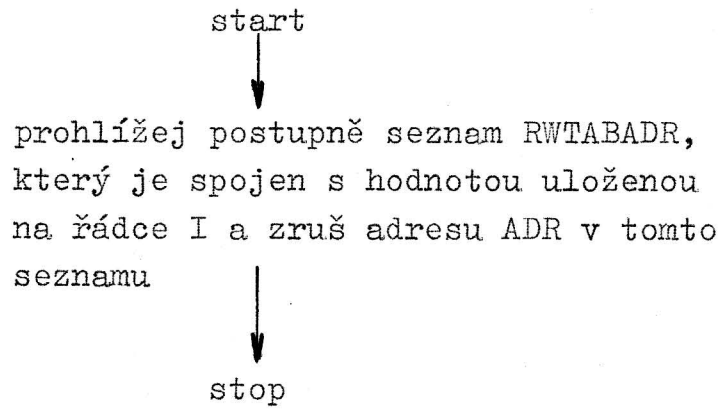
ADRINTO (ref[ ]int A,ref [,] int RWTABADR,ref int ADR,CAF)void:



DELETENFD (ref nfd NFD, int I) void:



DELETENFDADR (ref nfd NFD, int ADR, I) void:



6.5.3. Operace nad oblastí reprezentující relace

Uvedeme jména podprogramů realizujících operace nad oblastí reprezentující relace a jejich vývojové diagramy. Podrobné vývojové diagramy jsou uvedeny v příloze D.

INITREL - zajistí vygenerování potřebného paměťového prostoru pro uložení hodnot a provede prvotní nastavení hodnot v tabulkách.

INSERTREL - zajistí uložení dané n-tice do relace, pokud tato n-tice v relaci ještě není založena. Hodnota jednotlivých oblastí n-tice jsou uloženy do příslušných nefundamentálních oblastí nebo do fundamentálních oblastí, pokud existují.

DELETEREL - zajistí zrušení n-tice v relaci, tj. zruší se příslušné hodnoty v oblasti reprezentující relace a v nefundamentálních oblastech.

TISKREL - zajistí vytištění hodnot uložených v oblasti reprezentující relace. Podprogram slouží jen pro účely ladění.

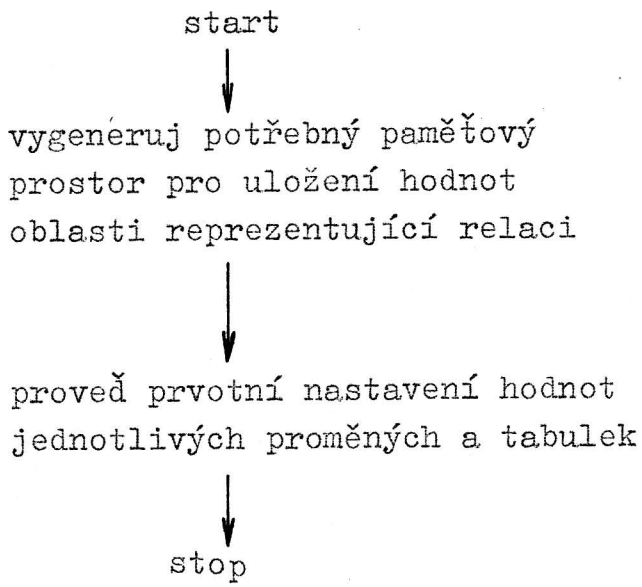
TISKRELALL - zajistí se vytištění hodnot uložených v oblasti reprezentující relace, v nefundamentálních oblastech a ve fundamentálních oblastech, pokud existují. Podprogram slouží jen pro účely ladění.

FINDREL - zjistí, zda daná n-tice je již v relaci uložena. Pokud ano, pak je vrácen index řádky tabulky RWTABZA, kde je n-tice uložena.

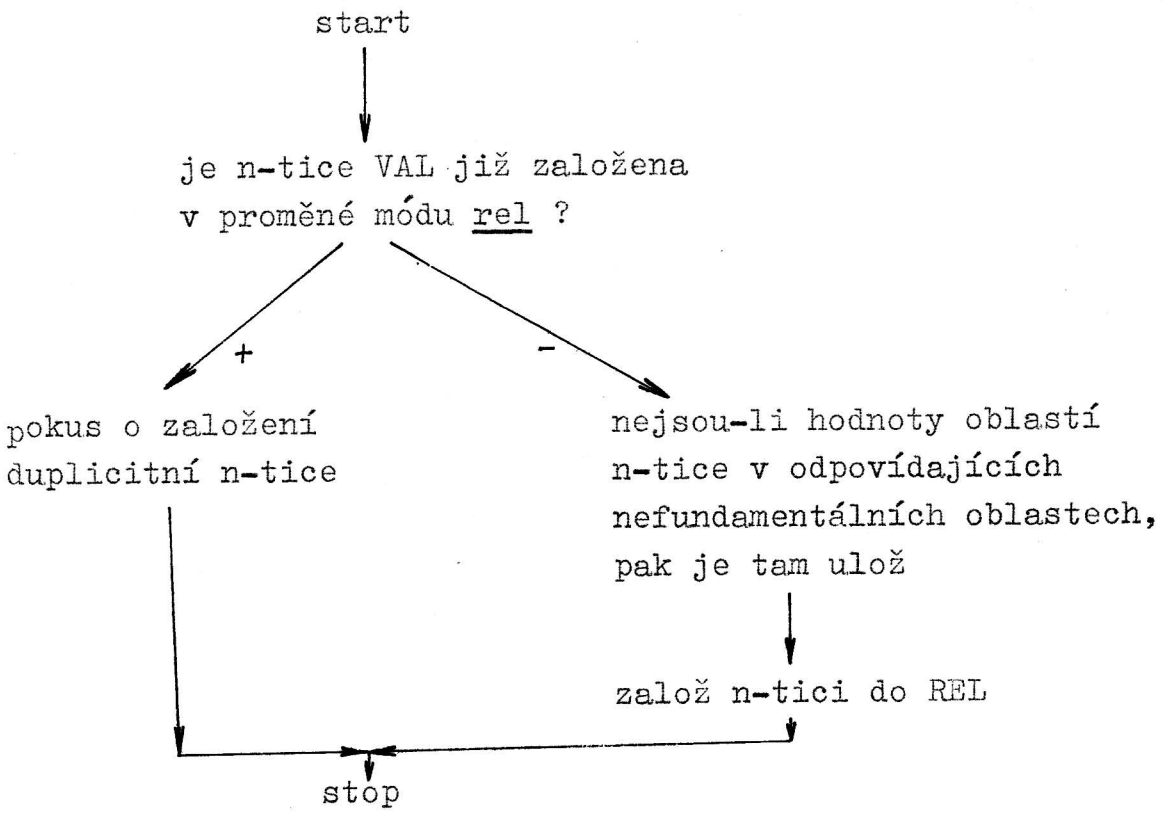
ADDSPACEREL - zajistí přidání paměťového místa proměné módu rel, tj. oblasti reprezentující relace.

SORTREL - zajistí přeuspořádání oblasti reprezentující relace. V některých případech vyvolá podprogram PŘEINDEXUJRWTABADR.

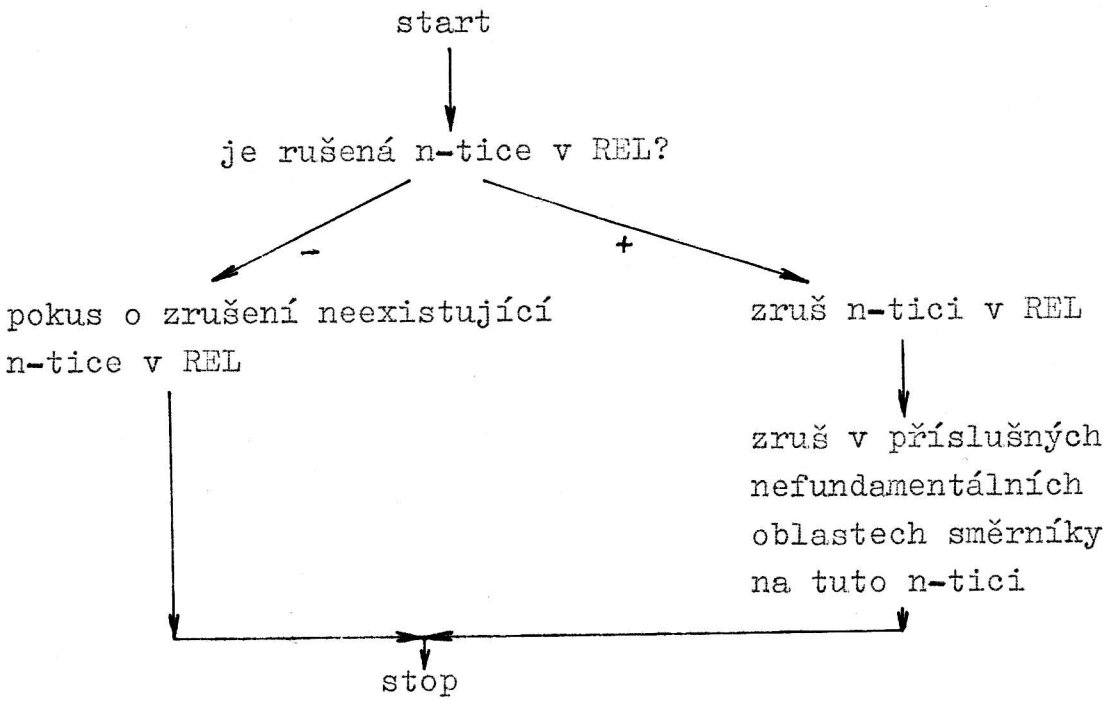
INITREL (ref rel REL, [ ] ref nfd DOM, int C1) void:



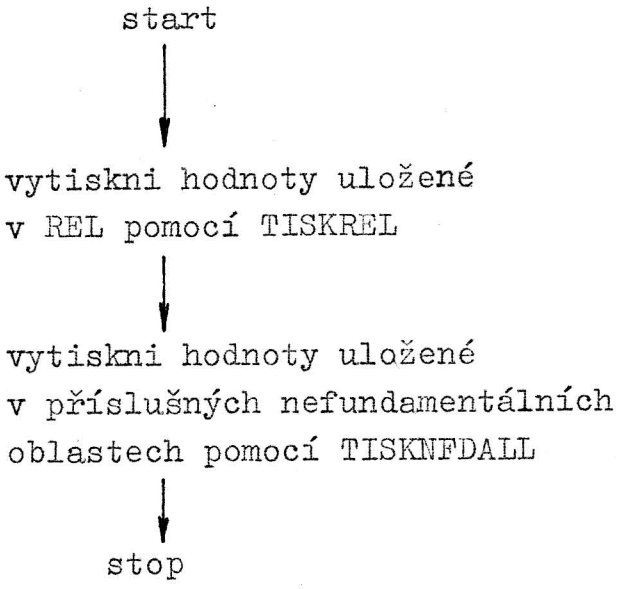
INSERTREL (ref rel REL, [ ] string VAL) void:



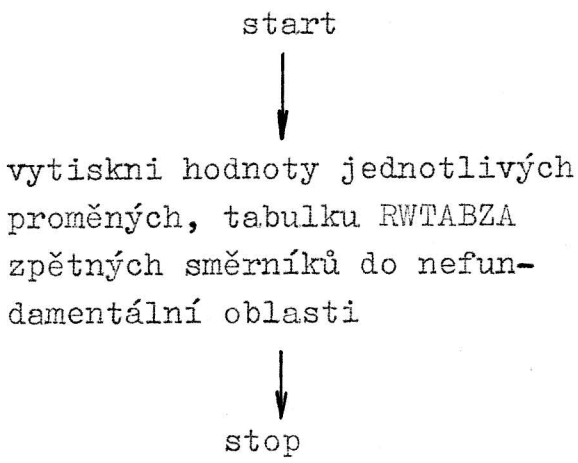
DELETEREL (ref rel REL,[ ] string VAL) void:



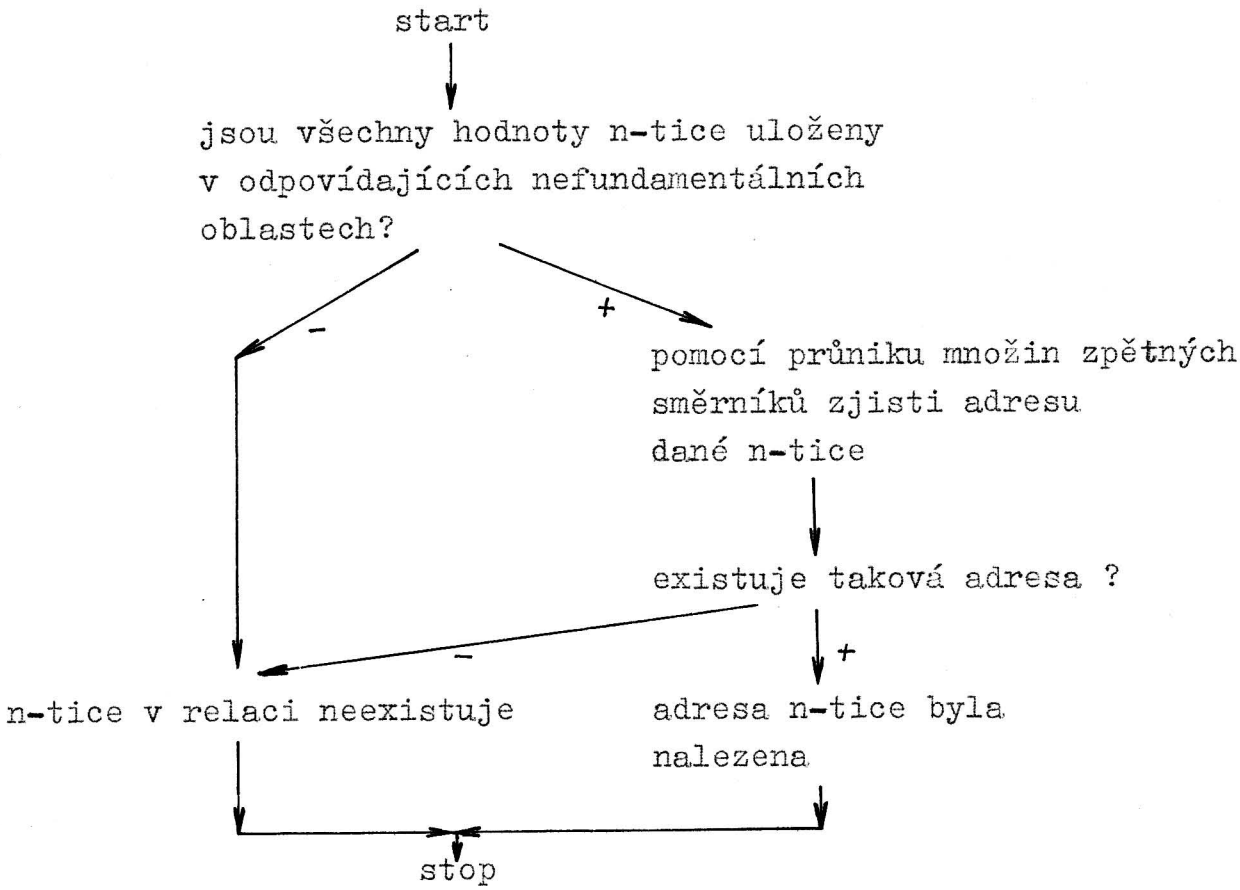
TISKRELALL (ref rel REL) void:



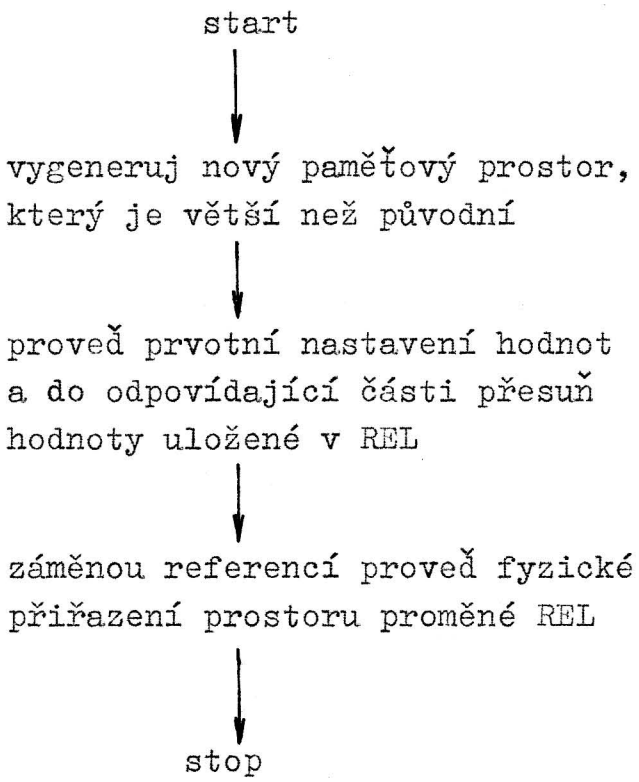
TISKREL (ref rel REL) void:



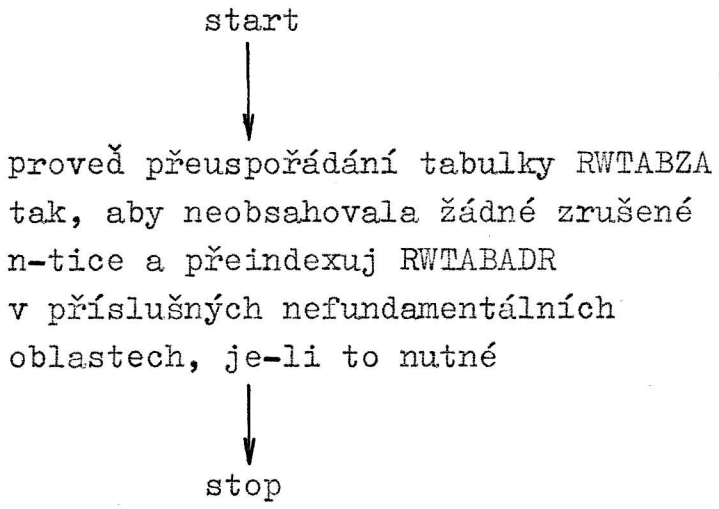
FINDREL (ref rel REL,[ ] string V, ref bool BOOL,ref,int L)void:



ADDSPACEREL (ref rel REL) void:



SORTREL (ref rel REL) void:



Operace typu sjednocení, průniku atd. přímo s relacemi jsou velmi náročné na čas, který je nutný k založení relace a vložení příslušných n-tic. Tato nevýhoda nejvíce vynikne v takových případech, kdy se vytváří mezivýsledky, jež jsou relacemi. Takový příklad je uveden např. v kap.5., kde R1 a R2 jsou relace dočasné platnosti. Proto byla hledána taková struktura, která by sloužila k rychlému uložení mezivýsledků. Taková datová struktura byla nalezena a její mód byl nazván cursor. Její použití a význam bude uveden dále.

7. Datová struktura pro uchovávání mezivýsledků

Vzhledem k tomu, že není možné zakládat relaci jako mezivýsledek především z časových důvodů, byla vytvořena datová struktura, která je vhodná pro ukládání mezivýsledků a pro určitý stupeň optimalizace příkazu nebo dotazu uživatele. Navrženou datovou strukturu lze v Algolu 68 popsat takto:

```
mode cursor = struct (ref cursor A,B,
                      ref rel C,D,
                      ref[ ]int ADR, INNFD,
                      [1:2] int CODE ) ;
```

obr.7.1.

Význam jednotlivých proměných:

CODE - slouží k zakódování, zda je použito konstrukce ref rel ...., nebo zda je použito ref cursor ...., k identifikaci, jaké operace bylo použito při vytváření proměné atp. Tímto způsobem se vyhneme konstrukci union.

ADR - slouží k uložení adres již určených n-tic relace.

INNFD - slouží k uložení indexů, které identifikují určitou nefundamentální oblast.

A,B,C,D - určují na jaké relace nebo mezivýsledky je proměná módu cursor vázána.

Nyní lze definovat operace pomocí operátorů, jejichž definice v Algolu 68 má tvar:

1) op . = (ref rel A, int B) ref nfd:

Operátor slouží k vybrání příslušné nefundamentální oblasti. Uvažujeme-li relaci A(B,C,D,E,F), pak A.2 má stejný význam jako C, tj. určuje nefundamentální oblast C a relace A.

2) op { <= , < , = , > , >= } = (ref nfd A,[ ]char B) ref cursor:

Operátory slouží k výběru n-tice relace pomocí podmínky, která je kladena na hodnoty zvolené nefundamentální oblasti dané relace, např. A.2 = "3780". V určitých případech není nutné určovat, že jde o relaci A. Proto byly výše uvedené operátory ještě definovány takto:

op { <= , < , = , > , >= } = (int A, [ ]char B) ref cursor:

Poznámka 1 :

Závorky { } jsou použity jen pro zkrácení zápisu, aby nebylo nutné rozepisovat definici operátoru pro každý typ.

3)

op { and / or } = (ref cursor A,B) ref cursor :

Operátory slouží ke spojování podmínek kladených např. při dotazu. Priority operací jsou stejné jako u operací s booleovskými hodnotami. Pořadí prováděných operací lze změnit pomocí závorek, např.

A.2 <= "3700" and (A.1 < = "700" or A.4 > = "1780")

4)

op : = (ref rel A, ref cursor B) ref cursor :

Operátor slouží k výběru podle kritéria, např.

A: (A.2 <= "700" or A.1 > = "1500")

Prioritu operátoru je možno zvolit tak, aby nebylo nutno používat závorek. Vzhledem k tomu, že můžeme někdy požadovat, aby výběr byl aplikován na mezivýsledek, tj. na proměnou módu cursor, je operátor výběru ještě definován takto:

op: = (ref cursor A,B) ref cursor:

5)

op % = (ref rel A, [ ]int B) ref cursor:

Operátor zajišťuje projekci relace A na oblasti, jejichž indexy jsou uvedeny ve vektoru B. V tomto případě se používá poziční identifikace oblastí. Použijeme-li operátoru ".", lze identifikovat oblasti pomocí jména oblastí. Při běžném používání operací s relacemi může uživatel požadovat projekci mezivýsledku na určité oblasti. Proto byl ještě definován operátor projekce takto:

```
op % = (ref cursor A, [] int B) ref cursor:
```

Pak lze např. napsat příkaz:

```
A% (1,2,3,5)%(1,4,2)
```

6)

```
op + = (ref rel A,B) ref cursor:
```

Operátor zajišťuje sjednocení dvou relací. Někdy však může být požadováno sjednocení mezivýsledků, tj. proměných módu cursor a nebo proměné módu rel a cursor. Proto operátor + musí být ještě definován takto:

```
op + = (ref rel A, ref cursor B) ref cursor:
op + = (ref cursor A, ref rel B) ref cursor:
op + = (ref cursor A,B) ref cursor:
```

Nyní lze napsat např. příkaz:

```
A: (A.1 <= "370")+A: (A.3 >= "4700" and A.4 <= "100")
```

7)

```
op * = (ref rel A,B) ref cursor:
```

Operátor zajišťuje průnik dvou relací. Pro běžné použití je nutné ještě definovat operátory :

```
op *= (ref rel A, ref cursor B) ref cursor:
op *= (ref cursor A, ref rel B) ref cursor:
op *= (ref cursor A,B) ref cursor:
```

8)

op - = (ref rel A,B) ref cursor:

Operátor zajistí rozdíl dvou relací. Je opět nutné definovat operátor i pro následující případy:

op - = (ref rel A, ref cursor B) ref cursor:

op - = (ref cursor A, ref rel B) ref cursor:

op - = (ref cursor A,B) ref cursor:

9)

op : = = (ref rel A, ref rel B) ref rel:

Operátor zajistí přiřazení relací, tj. vytvoření kopie A relace B.

10)

op := = (ref rel A, ref cursor B) ref rel:

Operátor zajistí skutečné vytvoření nové relace, která má hodnoty určené proměnou módu cursor, např.:

B:= (A:(A.1 > = "3750")+A:(A.2 <= "3654"))%(1,3)

11)

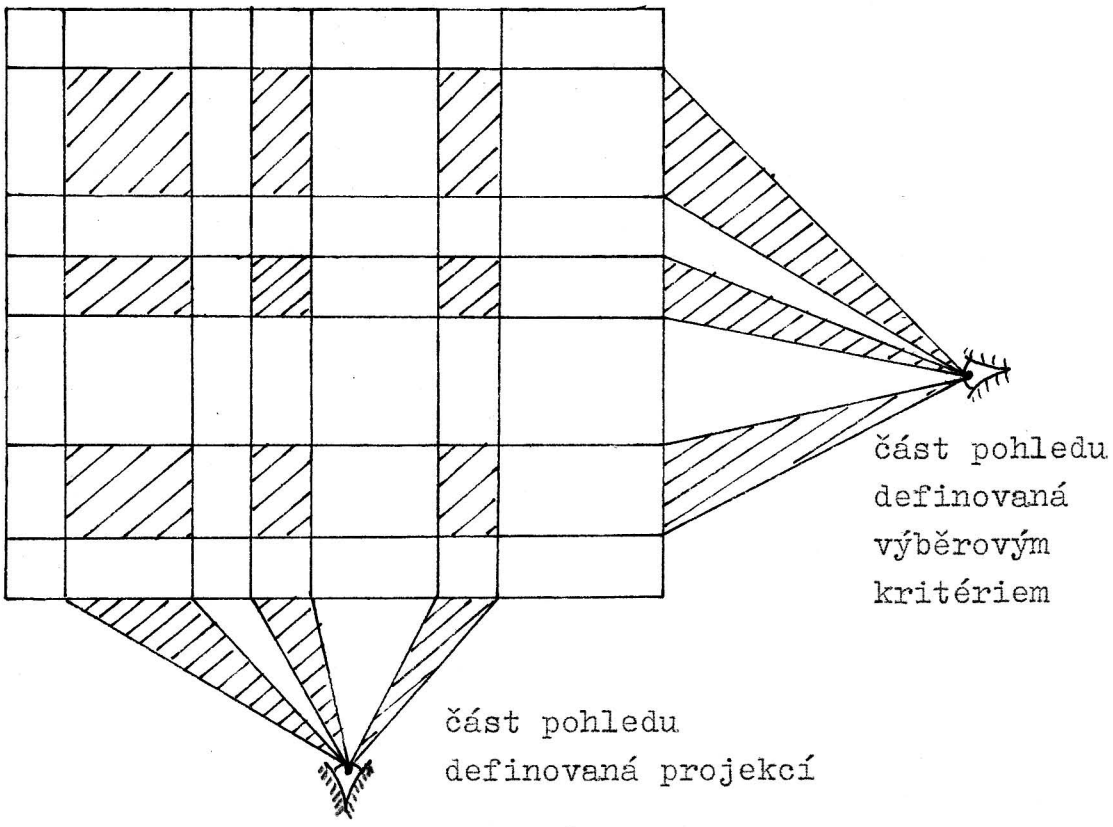
op := = (ref cursor A, ref cursor B) ref cursor:

Operátor zajistí vytvoření kopie proměné módu cursor. Dále uvidéme, že tato operace je vhodná pro pohled na relaci, jak byl uveden v kap.4. pod názvem view. Jak si lze představit pohled na relaci, je ukázáno na obr.7.2.

Předložená datová struktura proměné módu cursor umožňuje částečně optimalizovat požadavky uživatelů, zejména v případech požadování projekce na určité oblasti relace a při formulaci výběru, neboť nejsme nuceni v mezikrocích fyzicky zakládat pracovní relaci k uložení mezivýsledku. Optimalizaci v případě projekce si ukažme na příkladě. Uvažujme následující dotaz uživatele:

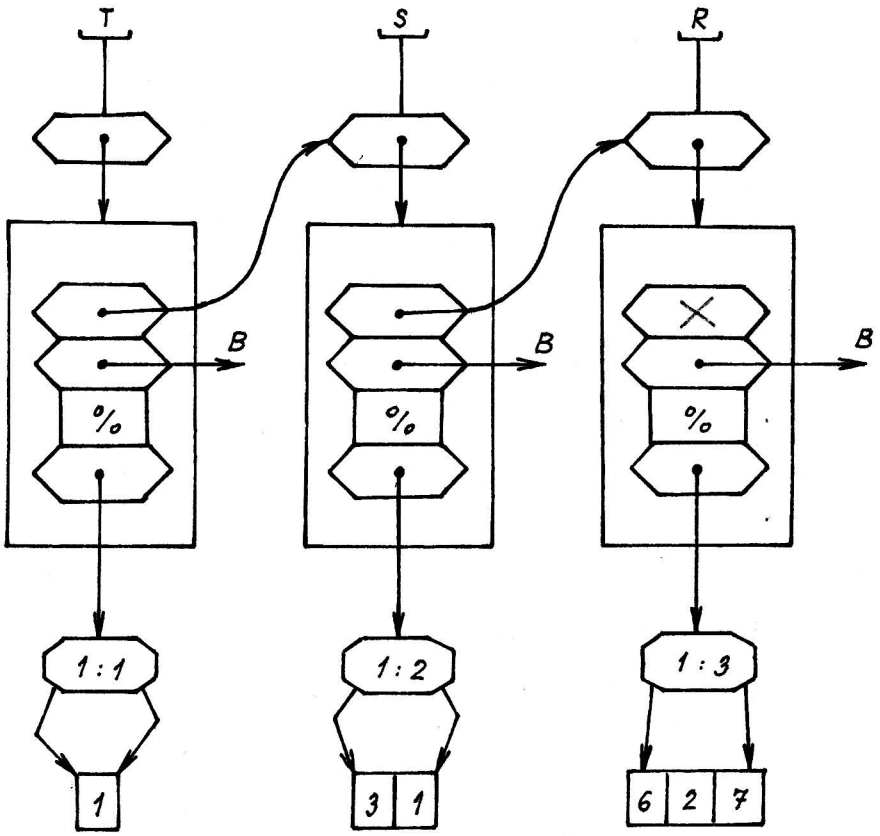
E:= B% (6,2,7)%(3,1)%(1),

kde indexy jsou čísla příslušných oblastí, které požadujeme, relace B. Vydání výše uvedeného příkazu způsobí vygenerování datové struktury, která je ukázána na obr. 7.3. Pro jednoduchost jsou zakresleny pouze části nutné k pochopení.



Pohled na relaci jako "dynamické" okno na relaci definované projekcí a výběrovým kritériem

obr.7.2.



Datová struktura příkazu vícenásobné projekce před optimalizací

obr.7.3.

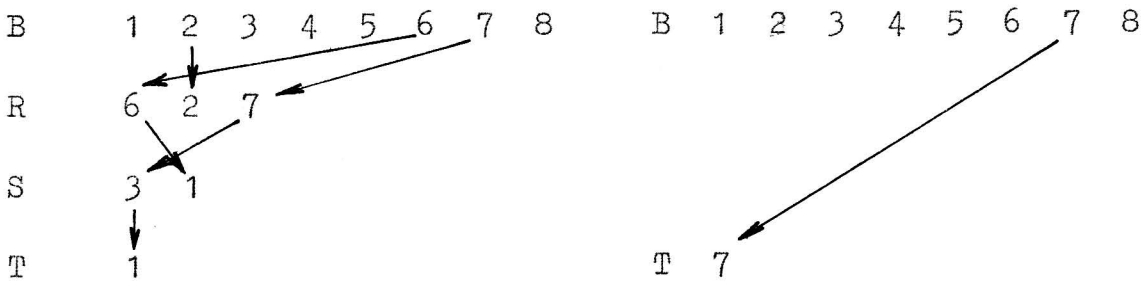
Proměné R,S,T z obr.7.3. jsou proměné módu cursor. Jsou to tedy mezivýsledky, jež jsou definovány takto:

R = B% (6,2,7)

S = R% (3,1)

T = S% (1)

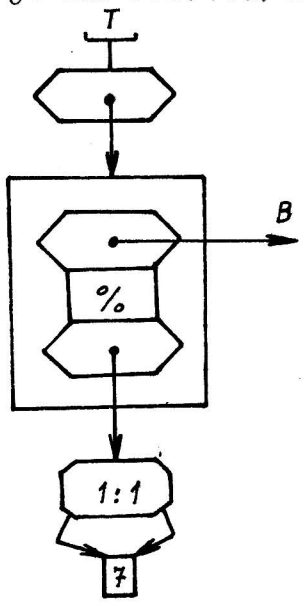
Požadavek projekce s příslušnými mezikroky lze znázornit takto:



T:=B%(6,2,7)%(3,1)%(1) T:=B(7)

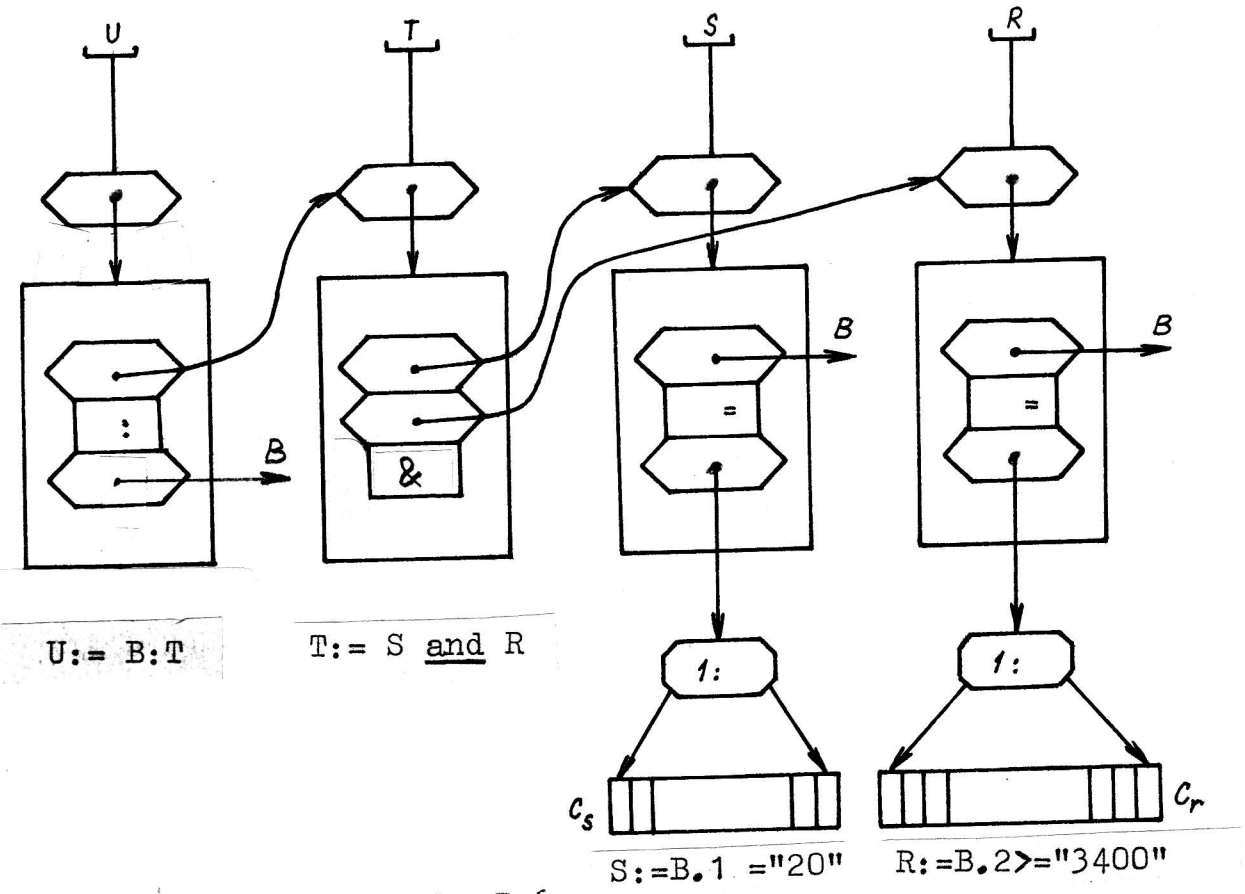
obr.7.4.

Z obr.7.4. vyplývá, že mezivýsledek T může být získán přímo, a to projekcí T:=B%(7). V případě, že bychom posloupnost projekcí zkrátili na jednu projekci, zrychlili bychom podstatným způsobem vykonání daného příkazu uživatele. Po optimalizaci bychom místo datových struktur ukázaných na obr.7.3. dostali datovou strukturu, která je na obr.7.5, a která je podstatně jednodušší.



Optimalizovaná datová struktura projekce

obr.7.5.



obr.7.6.

V případě výběru n-tic z relace máme pak možnost pracovat pouze s indexy n-tic. Uvažujme následující výběr:

A:= B:(B.1 < = "20" and B.2 > = "3400"),

kde A a B jsou "vhodné" relace. Pak dotaz způsobí vytvoření datové struktury, která je uvedena na obr.7.6. Proměné U,R,S,T jsou proměné módu cursor, které mají omezenou existenci a které existují až do okamžiku, kdy se provede přiřazení relaci A. Výhodou je, že při operacích and a or pracujeme pouze s indexy n-tic příslušné relace (vektor C_r,C_s na obr.7.6). Je-li C_s nebo C_r vektor indexů vybraných n-tic nulové délky, pak A je prázdnou relací. Navíc je dodržován požadavek jedinečnosti výskytu n-tice v relaci, jsou-li operace průniku, sjednocení a rozdílu množin indexů vybraných n-tic vhodně realizovány.

Proměnou módu cursor lze navíc ještě použít k reprezentaci pohledu na bázi dat, a to tím způsobem, že na levé straně přiřazení je proměná módu cursor místo relace. Příklad ukazující pohled na relace je uveden v příloze H.

8. Optimalizace příkazů uživatele

Relační databázový systém poskytuje uživateli spolu s tabulkovým pohledem na data také pohled, který je nezávislý na počítači nebo na implementaci. Uživatel nemusí znát nic o implementaci při formulování svého dotazu a nemůže být tedy ani odpovědný za efektivitu, se kterou je dotaz zodpovězen. Složitý dotaz na rozsáhlou část dat uložených v bázi dat je zodpovězen za určitou dobu, která by neměla kritickým způsobem záviset na způsobu dotazu nebo jeho formulaci. Cíl dotazu, tj. výsledek dotazu, musí být správný, ale jednotlivé části dotazu mohou být transforomovány na jiné ekvivalentní dotazy, abychom zajistili co nejrychlejší odpověď. Náhodní uživatelé by pak byli ochráněni od katastrofálně neefektivních dotazů. Současně by neměl být zkušený uživatel, který umí vyjádřit svůj dotaz efektivně nebo nepožaduje optimalizaci, zatěžován náklady, které jsou spojeny s optimalizací. V předchozí kapitole jsme se zabývali pouze nejzákladnějšími způsoby optimalizace příkazů uživatele.

Transformace, kterými můžeme podstatným způsobem zlepšit efektivitu provádění příkazů uživatele, lze rozdělit do dvou skupin, a to:

1) změna pořadí operací, jež mají být provedeny tak, že obdržený výsledek bude odpovídat netransformovanému příkazu,

2) rozpoznávání společných subvýrazů a jejich oddělené vyhodnocování.

Správné tranformace pro změnu pořadí operací mohou být vyjádřeny pomocí algebraických zákonů, které vyjadřují, jaké příkazy jsou navzájem ekvivalentní. V druhém případě musíme být schopni rozpoznat, kdy jsou sekvence výrazů algebraicky ekvivalentní. Problémem pak je vybrat takovou posloupnost výrazů, která je algebraicky ekvi-

valentní původním příkazům uživatele. Naším úkolem je tedy vybrat takovou posloupnost výrazů, kterou lze nejefektivněji vyčíslit. Základním požadavkem pak je, abychom byli schopni určit cenu vyčíslení daného výrazu. V [24] je ukázáno, že takový odhad ceny vyčíslení výrazu je extrémně obtížnou záležitostí.

Vzhledem k obtížnosti určení ceny vyčíslení výrazu je vhodné se vyhnout transformacím , u nichž rychlost závisí kritickým způsobem na nějaké proměnnné vlastnosti, např. na kardinalitě relace. Dále budou uvedeny takové transformace, které vždy zaručují vyšší efektivitu při vyhodnocování výrazu. Pro představu je vhodné pohlížet na dotazy jako na stromy, kde uzly stromu jsou reprezentovány symboly operací a listy stromu jsou pak relace, hodnoty, identifikátory oblastí relací nebo seznamy oblastí, na něž se má provést projekce.

8.1. Optimalizace posloupnosti projekcí a výběrů

Nejjednodušší transformací je přechod od posloupnosti projekcí k projekci jednoduché a od posloupnosti výběrů k výběru jednoduchému.

Posloupnost výběrů může být nahražena takto:

$$A\,\%\,T_1\,\%\,T_2\,\%\,T_3\,\%\,\ldots\,\%\,T_{n-1}\,\%\,T_n = A\,\%\,T,$$

kde $T_1,\ldots,T_n$ jsou seznamy projekcí,

T je seznam projekce, který je vytvořen ze seznamů projekcí $T_1,\ldots,T_n$.

Každý seznam projekce má tvar:

$$T_k = (i_{k1}, i_{k2},\ldots,i_{kn_k}),$$

kde i_{kj} je celé číslo označující index j-tého komponentu seznamu T_k.

Příklad 1.

Uvažujme relaci B, která má osm oblastí, Pak lze ukázat, že platí:

$$B\ \%(6,2,7)\%(3,1)\%(1) = B\ \%(7)$$

viz též obr.7.4.

Obecný algoritmus redukce počtu projekcí redukuje vždy dvě projekce do jedné a od poslední, tj. co leží nejvíce vpravo, k první. Algoritmus redukce lze popsat takto:

```
for K from N to 2 by -1 do
    for J to upb T_k do
        T_k[J] := T_{k-1} [ T_k [J] ]
    od
od
```

Použijeme-li výše uvedený příklad k ilustraci funkce tohoto algoritmu, pak mezikroky jsou:

B %(6,2,7)%(3,1)%(1)	původní výraz
B %(6,2,7)%(3)	výraz po 1.kroku algoritmu redukce
B %(7)	výraz po 2. kroku algoritmu redukce a také výsledek algoritmu

Další jednoduchou transformací je náhrada posloupnosti výběrů jedním výběrem. Posloupnost může být nahrazena takto:

$$A: F_1 : F_2 : \ldots : F_n = A : (F_1 \& F_2 \& \ldots \& F_n)$$

Je vhodné zde poznamenat, že předpokládáme, že výběr lze provést efektivněji pro složené kritérium, než při vyhodnocování jednoduchých výběrových kritérií, které jsou kladeny na mezivýsledky. Ve skutečnosti se výběr bude provádět tak, jak je ukázáno níže:

$$A : (F_1 \& F_2 \& \ldots \& F_n) = A: F_1 \cap A:F_2 \cap \ldots \cap A:F_n$$

Je vhodné také připomenout, že v některých implementacích může být právě opak pravdou, tj. je výhodnější vybírat pomocí posloupnosti výběrových kritérií.

8.2. Optimalizace výrazů pomocí algebraických zákonů

Jde o více složité transformace, které umožňují zrušení prázdných relací a redundantních operací, jako $A \cap A$, $A - A$ apod. Na první pohled se zdá, že jde o zcela triviální záležitost, ale ve skutečnosti jde o velmi obtížnou identifikaci společných subvýrazů, která je následována aplikací různých algebraických zákonů, které ruší neužitečnou nadbytečnost. V tab. 8.1. jsou uvedeny základní algebraické zákony, které lze aplikovat na relace.

výraz	redukováno na	
$A \cup \emptyset = \emptyset \cup A$	A	
$A \cap \emptyset = \emptyset \cap A$	$\emptyset$	
$A - \emptyset$	A	
$\emptyset - A$	$\emptyset$	
$\emptyset \,\%\, T$	$\emptyset$	
$\emptyset : F$	$\emptyset$	
A : true	A	
A : false	$\emptyset$	
$A \times \emptyset, \emptyset \times A$	$\emptyset$	
$A \cup A$	A	Idempotentní zákony relační algebry
$A \cap A$	A	
$A - A$	$\emptyset$	

tab. 8.1.

I když se zdá, že aplikace výše uvedených algebraických operací je jednoduchou záležitostí, jde ve skutečnosti o velmi složitý proces, který lze demonstrovat na jednoduchém příkladě.

Uvažujme následující dotaz uživatele:

$$((B \cup C) - (C \cup B)) \cap B,$$

který lze redukovat na prázdnou relaci. Vidíme, že jde o rozpoznávání společných subvýrazů.

Podobně jako pro operace s relacemi jsou známy zákony Booleovy algebry, které jsou uvedeny v tab.8.2., jejichž aplikací lze někdy podstatným způsobem zjednodušit výběrové kritérium ve výběru.

výraz	redukováno na
B & true = true & B	B
B & false = false & B	false
B ∨ true = true ∨ B	true
B ∨ false = false ∨ B	B
B & B	B
B ∨ B	B
B & ¬B	false
B ∨ ¬B	true

tab.8.2.

Rozpoznávání společných subvýrazů lze nalézt např. v [24]. Algoritmus pro optimalizaci s využitím zákonů uvedených v tab.8.1. a 8.2. lze zapsat takto:

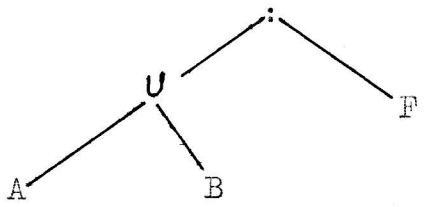
Začni od listů stromu a postupuj směrem nahoru rušením společných subvýrazů a aplikováním algebraických zákonů z tab.8.1. a 8.2.

8.3. Optimalizace pomocí přesunů výběrového kritéria ve výběru

V této části budeme vycházet z předpokladu, že výběr podstatným způsobem redukuje kardinalitu relace. Budeme se snažit přesouvat výběry co možná nejníže až na původní relace. Uvažujme příklad:

$$(A \cup B) : F,$$

který je na obr. 8.1.



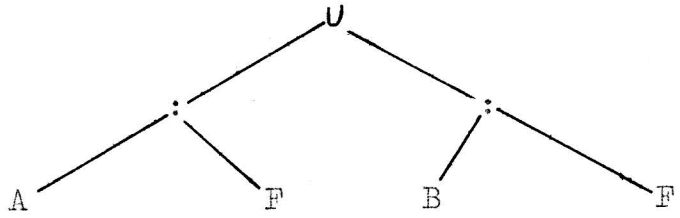
(A ∪ B) : F

obr.8.1.

Optimalizací výrazu přesunutím výběrového kritéria až k relacím pak dostáváme výraz:

(A : F) ∪ (B : F),

který je zobrazen na obr.8.2.



obr.8.2.

Pro každou operaci existuje pravidlo, jak přesunout výběrové kritérium směrem k listům ve stromě, tj. k relacím. Tato pravidla jsou shrnuta v tab.8.3.

výraz	převedeno na	alternativa
$(A : F) : G$	$A : (F \,\&\, G)$	
$(A \cup B) : F$	$(A : F) \cup (B : F)$	
$(A \cap B) : F$	$(A : F) \cap (B : F)$	$(A:F) \cap B, A \cap (B:F)$
$(A - B) : F$	$(A : F) - (B:F)$	$(A:F) - B$
$^{*}(A \,\%\, T) : F$	$(A : F) \,\%\, T'$	
$^{+}(A * B) : F$	$((A:F_1) * (B * F_2)):F_3$	

tab. 8.3.

*) T' je seznam projekce odvozený z původního seznamu T analogickým postupem jako v kap.8.1.

+) F je nutné transformovat do stavu $F_1 \,\&\, F_2 \,\&\, F_3$, kde: F_1 se týká pouze relace A,
F_2 se týká pouze relace B,
F_3 vyžaduje jak relaci A, tak i relaci B.

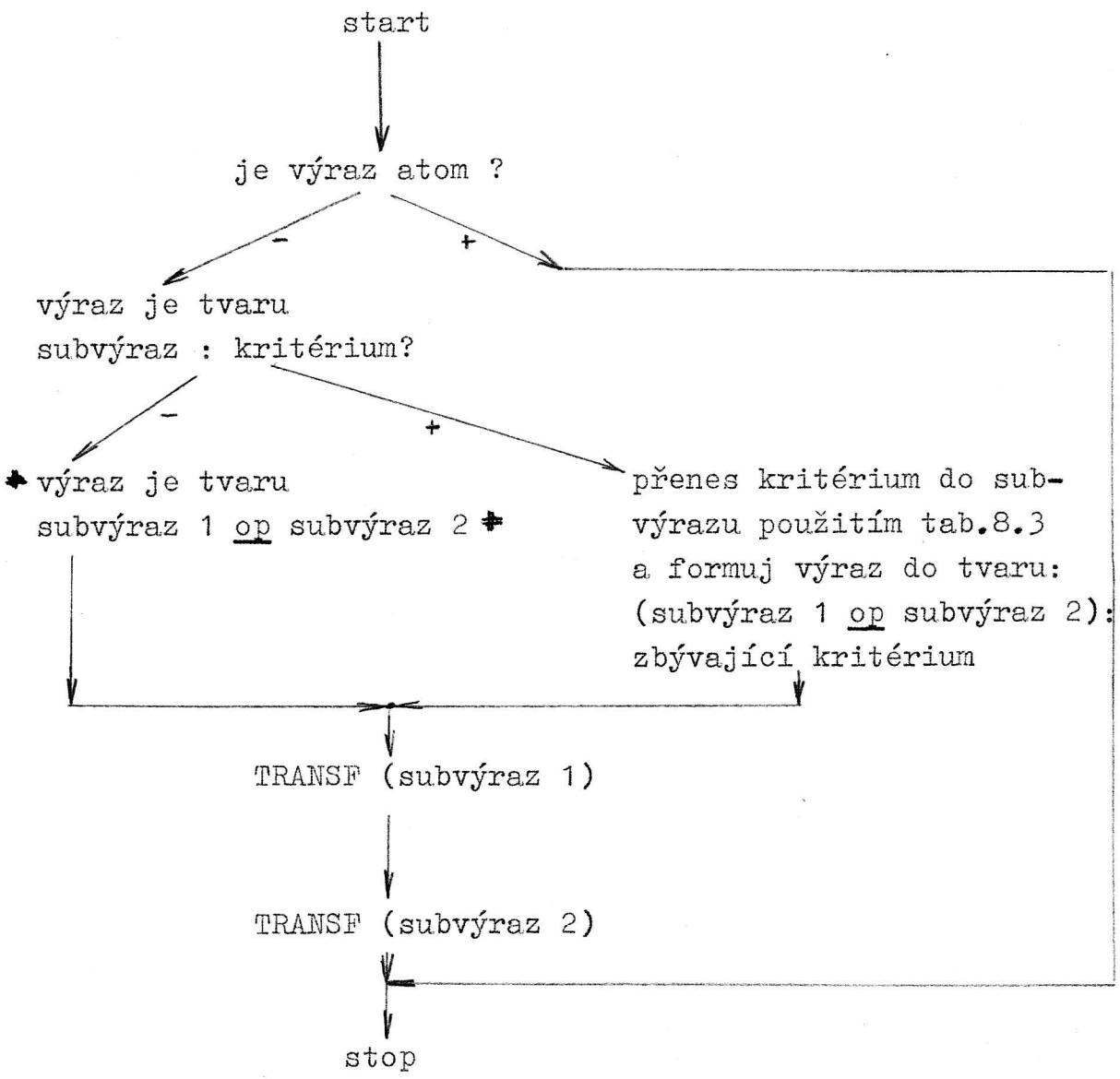
V tab.8.3. jsou navíc ukázány možné alternativy, které není vhodné používat, a to z následujících důvodů:

a) porušují symetričnost operace výběru a podtstatným způsobem ztěžují aplikaci idempotentních zákonů,

b) výběr obvykle podstatným způsobem snižuje kardinalitu relace, a proto by mělo být výhodnější výběr opakovat.

Algoritmus pro optimalizaci výrazu s výběrem lze popsat pomocí vývojového diagramu tak, jak je ukázáno na obr.8.3.

proc TRANSF = (ref exp výraz) void:



obr.8.3.

8.4. Účinnost optimalizačních postupů

Předložené principy optimalizačních postupů s určitými změnami, které byly vynuceny odlišným způsobem implementace relace, byly implementovány a ověřeny na PRTV systému [23] . Optimalizace byla realizována ve třech etapách, přičemž výsledkem každé etapy byl dokonalejší optimalizátor. V poslední etapě již bylo zahrnuto i vyčíslování společných subvýrazů odděleně.

Účinnost optimalizace dokládá tab.8.5, která byla převzata z [23] , představující spotřebu času na provedení sedmi různých dotazů. Pokus byl proveden na počítači IBM 370/145.

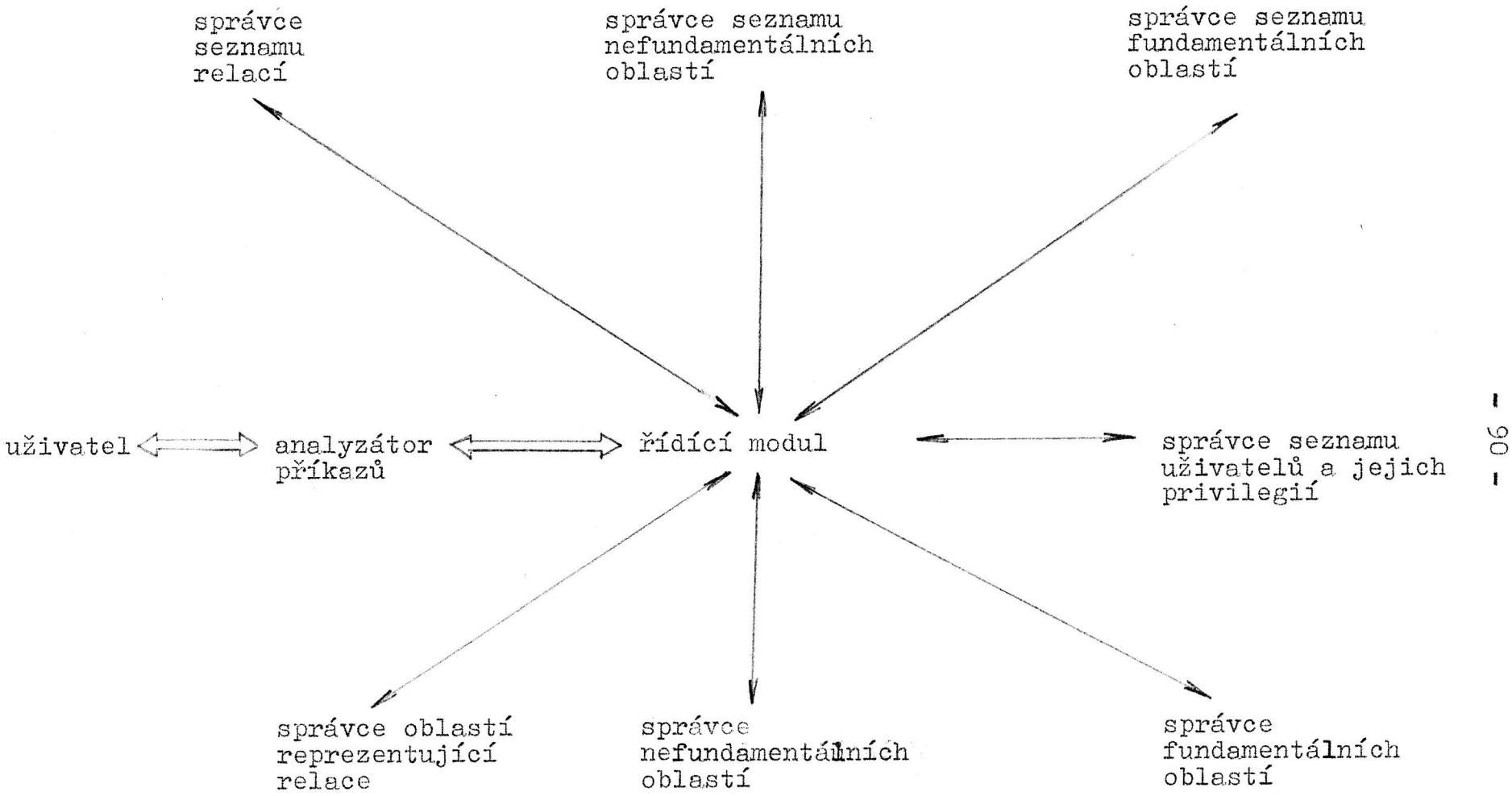
dotaz č.	spotřebovaný čas [s]			
	neoptimal.	mini optimal.	midi optimal.	maxi optimal.
1	2.01	2.21	0.73	0.81
2	10.37	1.20	1.69	2.22
3	3.78	2.73	2.74	2.95
4	5.49	5.74	5.83	8.41
5	4.13	4.36	3.60	3.76
6	3.78	4.12	4.25	3.23
7	asi 10 dní	asi 10 dní	1401.00	693.00

tab.8.5.

Z tabulky je patrné, že optimalizace dotazů uživatele má smysl a je vhodné se jí dále zabývat. Optimalizace dotazů uživatele bude předmětem další etapy státního úkolu III-2-2/2.

9. Filosofie systému řízení báze dat

U databázových systémů typu CODASYL je struktura báze dat určena před okamžikem jejího fyzického založení administrátorem báze dat. Jakákoliv změna struktury báze dat po fyzickém založení je obvykle velmi obtížná. U databázových systémů relačního typu se struktura báze dat mění dynamicky v čase, kdy jednotliví uživatelé zakládají nebo ruší relace. Požadavek dynamičnosti klade velké nároky na systém řízení báze dat. Na obr.9.1. jsou ukázány nejdůležitější moduly systému řízení relační báze dat. Jádro tvoří řídící modul, který řídí součinnost všech ostatních modulů, které zajišťují vykonávání určité skupiny příkazů. K správné činnosti řídící modul potřebuje informace o všech existujících relacích v bázi dat, o jejich vlastnících a o privilégiích, které byly vlastníky relací poskytnuty ostatním uživatelům. Informace o vlastnících relací a o privilégiích uživatelů jsou uloženy v seznamu uživatelů a jejich privilégií. Přístup k hodnotám uloženým v tomto seznamu je možný jen pomocí funkcí poskytnutých správcem tohoto seznamu. Problematika týkající se seznamu uživatelů a jejich privilégií je podrobně rozvedena např. v [7] ,[21] . Informace o existujících relacích jsou uloženy v seznamu relací, který je obhospodařován správcem seznamu relací. Podobnou funkci má také správce nefundamentálních oblastí a správce fundamentálních oblastí. Funkce modulů správců relací, fundamentálních a nefundamentálních oblastí spočívá také v tom, že uchovávají informace nejen, kde jsou příslušné hodnoty uchovávány, ale i tom, zda data jsou přístupná on-line či off-line, informace o rozměrech tabulek a o typech dat, jež uchovávají.

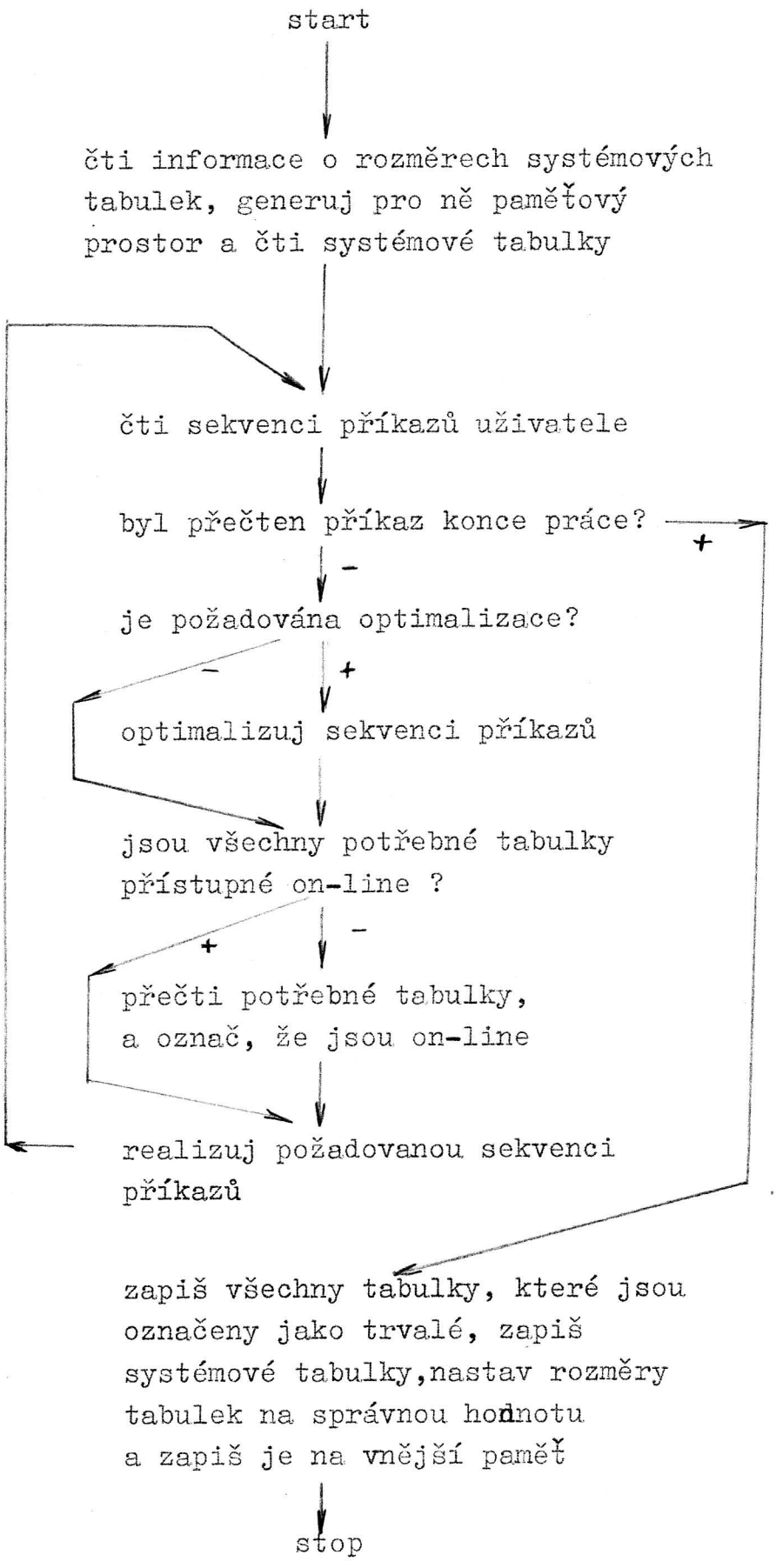

správce seznamu relací
správce seznamu nefundamentálních oblastí
správce seznamu fundamentálních oblastí
uživatel
analyzátor příkazů
řídící modul
správce seznamu uživatelů a jejich privilegií
správce oblastí reprezentující relace
správce nefundamentálních oblastí
správce fundamentálních oblastí

obr.9.1.

Výše uvedené moduly používají relací, které jsou organizovány ve tvaru tabulky a jsou sekvenčně prohledávány. Tyto "systémové" relace mají organizaci odlišnou od relací uložených v bázi dat, jejichž fyzická reprezentace byla vyložena v kap.6. Protože přístup do těchto relací není tak častý a kardinalita systémových relací není tak velká, lze použít sekvenčního vyhledávání se zanedbatelnou časovou ztrátou. V případě, že půjde o velmi rozsáhlou bázi dat, lze využít vlastnosti, že několik oblastí relace tvoří klíč, který jednoznačně identifikuje každou n-tici v relaci, k rychlejšímu vyhledání n-tice v relaci (jde stále o "systémové" relace).

Uživatel je s řídícím modulem spojen pomocí analyzátoru příkazů, který může zahrnovat i optimalizaci příkazů uživatele.

Jednotlivé akce systému jsou znázorněny na obr.9.2. Rozměry tabulek jsou generovány větší, aby bylo snadné přidávat do báze dat další relace. V případě, že tabulky budou již plné, zvětší se paměťový prostor automaticky při založení další relace do báze dat. V případě, že paměťový prostor je zaplněn, vyvolá se "carbage collector", který již nepotřebné relace uloží zpět na vnější paměť.


start
čti informace o rozměrech systémových tabulek, generuj pro ně paměťový prostor a čti systémové tabulky
čti sekvenci příkazů uživatele
byl přečten příkaz konce práce?
+
-
je požadována optimalizace?
-
+
optimalizuj sekvenci příkazů
jsou všechny potřebné tabulky přístupné on-line ?
+
-
přečti potřebné tabulky, a označ, že jsou on-line
realizuj požadovanou sekvenci příkazů
zapiš všechny tabulky, které jsou označeny jako trvalé, zapiš systémové tabulky,nastav rozměry tabulek na správnou hodnotu a zapiš je na vnější paměť
stop

obr.9.2.

10. Poznámky k fyzické realizaci

V předchozích kapitolách, hlavně v kap.6.,byla předložena nová koncepce implementace relace v relační bázi dat. Dosud jsme se otázkám týkajícím se spotřebě paměti vyhýbali tím, že jsme předpokládali virtuální paměť. Nyní naznačíme, jak relace implementovat na počítačích bez virtuální paměti. Uvidíme, že fyzická implementace nyní již zvládnutelnou záležitostí, neboť všechny logické vazby byly vyjasněny v předchozích kapitolách. Způsob fyzické implementace relace bude podobný způsobu, který byl použit v různých obměnách při implementaci báze dat typu CODASYL. Při implementaci báze dat typu CODASYL bylo použito stránek, které se postupně zaplňují příslušnými daty. V našem případě použijeme také stránek, které budou mít strukturu,již lze v Algolu 68 zapsat takto:

```
mode PAGE = struct (int NEXT, CURR, PRIO,
                    int PO1, PO2,
                   [1:t] char DATA,
                    int OVER ),
```

kde : CURR je číslo stránky,
PRIO je číslo předchozí stránky, pokud existuje,
NEXT je číslo následující stránky, pokud existuje,
OVER je číslo stránky "přetečení", pokud existuje,
PO1,PO2 jsou ukazatelé, kteří omezují neobsazenou zónu dat DATA shora a zdola,
t udává ve stránce počet Bytů, které jsou určeny k uložení dat.

Vezmeme-li v úvahu proměnou A módu fd, pak vidíme, že struktura obsahuje dvě tabulky proměnné délky, a to VAL a REF, kde tabulka REF představuje **přeindexování**, tj. vlastně simuluje hashovací funkci v rozptýlené organizaci.

Podobně je tomu u proměné módu nfd, kde vektor TAB [,3] vlastně také zastupuje hashovací funkci. Jediný problém, který zde zůstává je návrh takové hashovací funkce, která by rovnoměrně zaplňovala stránky přidělené fundamentální či nefundamentální oblasti. U proměné módu rel je otázka umístění na fyzické stránky záležitostí zcela triviální, neboť se jednotlivé řádky tabulky RWTABZA umisťují za sebe, a když se stránka zaplní, pokračuje se na stránce následující.

U "systémových" relací je situace obdobná jako při ukládání proměných módu rel. V systémových relacích pak bude místo odkazu na proměnou módu rel, nfd, fd pomocí ref jen číslo první stránky řetězu stránek, kde jsou uloženy proměné módu rel, nfd, fd (jedním z důvodů je to, že nelze ukládat reference na vnější paměť). Také u proměných módu rel bude []refnfd DOMAIN nahrazena řadou čísel prvních stránek řetězu stránek, kde jsou příslušné nefundamentální oblasti uloženy. Podobným způsobem musí být provedeny změny u nefundamentální oblasti.

Výhody předložené fyzické realizace:

1) snadná spolupráce s vnější pamětí - všechny stránky jsou stejně dlouhé,

2) snadné prodlužování tabulek, neboť není nutné přepisovat hodnoty do nově vygenerovaného paměťového prostoru, jak tomu bylo u návrhu, který byl proveden v kap.6. Stačí jen požádat o přidělení další stránky z "hromady" volných stránek,

3) jednoduché algoritmy na reorganizaci jednotlivých oblastí,

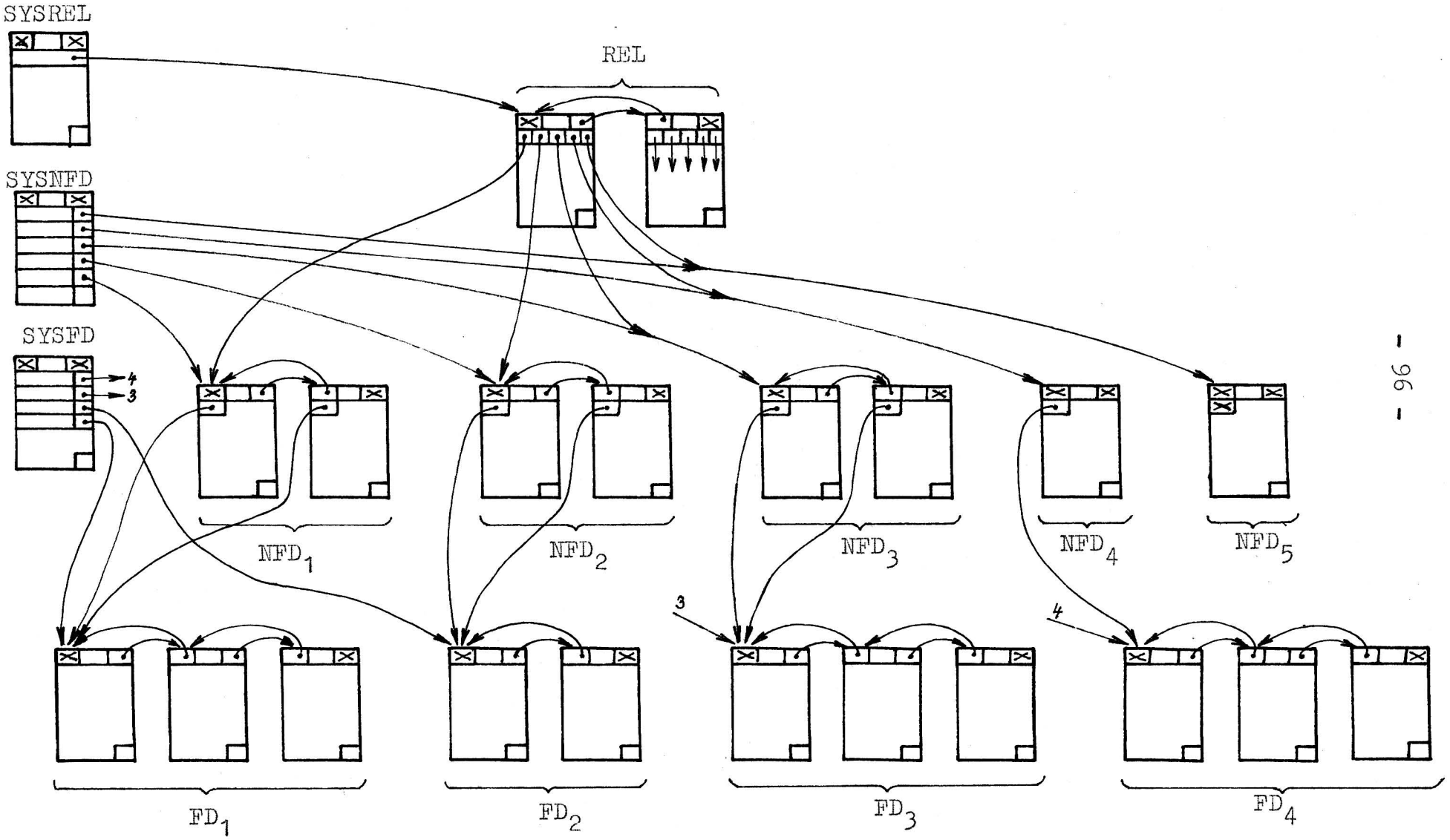
4) je možné mít stránky na více souborech a pro určitou oblast použití pak připojit on-line jen určité soubory, a tím i stránky. Pak nemusí být připojena celá báze dat on-line, což je obdoba pojmu AREA u databázových systémů typu CODASYL.

Nevýhody:

1) prodloužení doby odezvy na příkaz uživatele, neboť se musí uvažovat i čas periferních přenosů,

2) problém stanovení hashovací funkce při ukládání hodnot na stránky.

Na obr.10.1. je ukázáno, jak si lze představit fyzickou reprezentaci, přičemž SYSREL je seznam relací, které jsou v bázi dat uloženy. SYSNFD je seznam nefundamentálních oblastí, SYSFD je seznam fundamentálních oblastí.


SYSREL
REL
SYSNFD
SYSFD
4
3
NFD1
NFD2
NFD3
NFD4
NFD5
3
4
FD1
FD2
FD3
FD4

obr. 10.1.

11. Závěr

11.1. Zhodnocení zvoleného přístupu

V celé historii inženýrství lze hlavní zásady shrnout takto: "Velké inženýrství je jednoduché inženýrství". Myšlenky, které se stávají příliš těžkopádnými a nepružnými, musí být nahrazeny novějšími a koncepčně jasnějšími myšlenkami, které ve srovnání se staršími jsou efektivnější v jejich jednoduchosti. V oblasti bází dat musíme hledat takový způsob popisu dat a manipulace s daty, který může být snadno pochopen i uživateli bez znalosti programování, umožňující maximální pružnost ve formulování neočekávaných a spontálních dotazů na terminálech. To umožňuje právě relační báze dat založená na tabulkové představě relací.

V dostupné literatuře se autoři spíše zabývali otázkami týkajícími se logické úrovně báze dat, tj. návrhu dotazovacích jazyků, vlivu normalizačních postupů na strukturu báze dat apod. Informace o fyzické realizaci jsou téměř nedostupné, i když některé lze vydedukovat z popisu vnějšího chování existujících relačních databázových systémů.

Těžiště práce je právě v návrhu filosofie implementace relace v relační bázi dat. Byl předložen zcela původní, pokud je autorovi známo, způsob implementace relace. Výhody, kterých bylo dosaženo vůči existujícím databázovým systémům, jsou tyto:

a) není nutný pojem klíče, nevyužívá se vlastnosti klíče při vyhledávání n-tic v relaci,

b) je možné provádět výběr podle libovolné oblasti relace, přičemž výběr není prováděn sekvenčním prohledáváním relace,

c) při operacích s relacemi se pracuje s indexy n-tic,

d) je vždy dodržena podmínka jedinečnosti výskytu n-tice v relaci,

e) je-li více stejných hodnot v oblasti relace, je hodnota fyzicky uložena jen jednou,

f) možnost zavedení společného slovníku pro uložení hodnot více oblastí různých relací,

g) možnost definování "okna" na relaci a případně i na celou bázi dat,

h) snadná implementace proměnné délky hodnot uložených v oblasti,

i) možnost paralelního zpracování, např. při výběrech podle n primitivních kriterií lze spustit n paralelních procesů vyhledávání.

Předložený způsob má i své nevýhody, a to:

a) komplikovanější algoritmus vkládání n-tice do relace, což je způsobeno:
   1) kontrolami spojenými s požadavkem jedinečného výskytu n-tice v relaci,
   2) nutností vytvoření přístupových cest k právě založené n-tici,

b) potřebou virtuální paměti.

Nevýhodou ad b) lze odstranit tím, že se použijí při fyzické realizaci stránkovací algoritmy, jak bylo naznačeno v kap.10.

Lze očekávat, že navržený princip implementace relace bude výhodné použít v aplikacích, kde se bude vyžadovat extrémně krátká doba při dotazech a doba odezvy systému při vkládání a změnách nebude kritická (pro většinu databázových systémů je charakteristické, že doba vkládání je nepoměrně delší než doba výběru).

11.2. Konkrétní závěry pro praxi a další rozvoj vědy

V práci je předložena jen část filosofie implementace relační báze dat, a to filosofie implementace relace. Pro úplné řešení problému je nutné vést výzkum v dalším období následujícími směry:

a) utajení dat a integrita báze dat,
b) vazba báze dat na operační systém s provedením příslušných úprav v operačním systému,
c) sdílení báze dat více uživateli najednou,
d) optimalizace dotazů uživatele,
e) možnostmi zavedení paralelního zpracování,
g) možnostmi použití předložené filosofie při návrhu distribuované báze dat.

Je vhodné na tomto místě poznamenat, že jde o oblast problémů velmi širokou, neboť doby realizace databázových systémů se počítají na desítky až stovky člověko-roků. Nelze proto očekávat, že k fyzické realizaci dojde v dohledné době bez podstatně větší koncentrace lidí pracujících na implementaci.

Vzhledem k tomu, že je možné vytvořit dotazovací jazyk, který by byl velmi blízký jazyku přirozenému, a vzhledem k vlastnostem relačního databázového systému, tj. snadné použití báze dat náhodným uživatelem a možnosti paralelního zpracování, a tím i krátké doby odezvy na dotaz uživatele, lze očekávat uplatnění takového systému u složek zajišťujících obranu a bezpečnost státu. Výhodné použití by také bylo u velkých podniků, jako součást ASŘ-P, kde by řídící pracovníci měli možnost klást dotazy a získávat skutečně aktuální informace, čímž by bylo umožněno pružné řízení podniku.

Nesporně zajímavým směrem výzkumu by byla aplikace pamětí s adresováním podle obsahu (Content Adressable Store-CAS). Jde o zcela neobvyklý způsob realizace systému adre-

sování, kdy se na systém CAS automaticky ukládají referenční indexy na všechny záznamy báze dat, které jsou uloženy na standartních vnějších pamětech. Je pochopitelné, že systém CAS celý systém báze dat zjednoduší.

Vzhledem k dostupnému hardwaru (TESLA 200) a nespolehlivosti překladačů programovacích jazyků, nelze celek odladit vzhledem k rozsáhlosti programů.

12. Literatura

[1] Astrahan M.M., Griffiths P.P., Camberlin D.D.: "SEQUEL 2: A Unified Approach to Data Definition, Manipulation and Control", IBM J. of Research and Development, 1976, No. 2

[2] Astrahan M.M.: "Implementation of a Structured English Query Language", CACM, Vol. 18, No. 10, October 1975

[3] Bracchi G., Paolini P.: "Relational Models and Languages: A Tutorial", Sommer-Seminar Datenmodelle und Systement würfe für Datenbanken Systeme, Sept. 1974, St. Austin

[4] Bracchi G., Paolini P., Fedeli A.: "A Language for a Relational Data Base Management System", Proc. Sixth Annual Princenton Conference on Information Science and Systems, March 1972

[5] Cagan C.: Data Management Systems", Melville, Los Angeles 1973

[6] Chamberlin D.D.: "Relational Data-Base Management Systems", ACM Computing Surveys, Vol. 8, No. 1, March 1976

[7] Chamberlin D.D., Gray J.N., Traiger I.L.: "Views, authorization and locking in a relational data base system", National Computer Conference 1975

[8] Chvalovský V.: "Banky dat", Praha, SNTL 1976

[9] Codd E.F.: "A Relational Model of Data for Large Shared Data Banks", CACM, Vol. 13, No. 6, June 1970

[10] Codd E.F.: "A Data Base Sublanguage Founded on the Relational Calculus", Workshop on Data Base Description, San Diego, California, November 11-12, 1971

[11] Codd E.F.: "Further Normalization of the Data Base Relational Model", Proceedings of Courant Computer Science Symposia 6, New York City, May 24-25, 1971

[12] Codd E.F.: "Relational Completness of Data Base Sublanguages", Courant Computer Science Symposia 6, New York City, May 24-25, 1971

[13] Date C.J.: "An Introduction to Data Base Systems", Addison-Wesley Publishing Company, London 1977

[14] DATABANKA 77, Sborník přednášek z celostátního semináře, ČSVTS, dům techniky Ústí n. Labem, 1977

[15] DATABANKA 77, Zkušenosti s budováním ASŘ - SČ, ČSVTS, dům techniky Ústí n. Labem 1977

[16] DATABANKA 74, Sborník přednášek z celostátního semináře ČSVTS, dům techniky Ústí n. Labem, 1974

[17] DATABANKA 75, Sborník přednášek z celostátního semináře ČSVTS, dům techniky Ústí n. Labem, 1975

[18] Early J.: "Relational Level Data Structures for Programming Languages", Acta Informatica, Vol. 2, No. 4, Springer-Verlag 1973

[19] Eckerhard P.: "Studie matematického modelu - Organizace informační základny pro vnitroústavní řízení a jeho programové řešení", INORGA, Praha 1976

[20] Integrated Data Store, Manual fy General Electric, 1967

[21] Griffiths P.P., Wade B.W.:"An Authorization Mechanism for a Relational Database Systems", ACM trans. on Database Systems, Vol. 1, No. 3, Sept. 1976, pp.242-255

[22] Hall P.A.V.: "Factorization of Algebraic Expressions", Report IBM UKSC 0055, 1975

[23] Hall P.A.V.: "Optimization of a Single Relational Expression in a Relational Data Base System", IBM Report UKSC 0076, June 1976

[24] Hall P.A.V.: "Common Subexpression Identification in General Algebraic Systems", Report IBM UKSC 0060, November 1974

[25] Hasselmeier H., Spruth W.: "Data Base Systems", Proceedings, 1975, Springer-Verlag, 1976

[26] Havlík A.: "Návrh a realizace systému řízení a organizace dat", Diplomová práce 1977, KTK-VŠSE Plzeň

[27] INGRES-version 6.0, Reference Manual and Support Files, PDP 11/45, DEC, November 1977

[28] Kalinichenko L.A.: "Toward Data Description Language for Data Base Description", ed. Donque B.C.M., Nijssen G.M., North Holland Publishing Company, New York, 1975

[29] Knuth D.H.: "The Art of Computer Programming", Addison-Wesley, Vol. 1, 1975

[30] Langefors B.: "Theoretical Analysis of Information Systems", Report

[31] Lindsey C.H., Meulen S.G.: "Informal Introduction to Algol68" North-Holland Publishing Company, New York, 1973

[32] Lowe T.C.: "Design principles for an on-line information retrivial system", Ph.D.Diss., Pensylvania, Philadelphia, 1966

[33] Lyon J.K.: "An Introduction to Data Base Design", John Wiley and Sons, New York, 1971

[34] Mareš J., Štembera V.: "Operační systém GEORGE 3 pro programátory", RPM-Škoda VPC, Plzeň, 1975

[35] Mareš J.: "Makropříkazy operačního systému GEORGE 3", RPM Škoda VPC, Plzeň, 1977

[36] Mareš J.: "Inzerát na Algol 68", Publikace VPC Škoda, Plzeň, 1978

[37] Martin J.: "Computer Data-Base Organization", Prentice-Hall, New York, 1975

[38] Nijssen G.M.: "Data Structuring in the DDL and Relational Model", in Data Base Management ed. J.W.Klimbie and K.L.Koffeman, North-Holland Publishing Company, 1974

[39] Kořínek S., Nováček J., Skala V.: "Souhrná zpráva za první etapu řešení dílčího státního úkolu III-2-2/2", výzkumná zpráva 209-07-78, KTK-VŠSE, Plzeň, leden 1978

[40] Olle T.W.: "Data Structures and Storage Structures for Generalized File", Processing File 68

[41] Operační systém DOS-EC, všeobecný popis, KSNP Praha, 1975

[42] Pečený I., Skala V.: "Počítače JSEP", Socialistická Akadenie ČSR, Praha 1976

[43] PL/1 Dos-EC, KSNP Praha 1974

[44] Programovanie v jazyku SIMULA 67, SVTS dům techniky Žilina 1977

[45] Příručka programátora PL/1 DOS-EC, KSNP Praha 1975

[46] Reisner P.: "Use of Psychological Experimentation as an Aid to Development of a Query Language", IEEE Trans. on Software Engineering, Vol. 3, No. 3, 1977

[47] Scheber A., Šturc J.: "Relačný model báz dát", Informačné systémy, No. 3, 1973

[48] Senko M.E.: "The DDL in the Context of a Multilevel Structured Description: DIAM II with FORAL", Data Base Description, ed. Douque B.C.M., Nijssen G.M., North-Holland Publishing Company, 1975

[49] SIBAS - The Data Base System, Users manual, Norsk Data. - - Elektronikk, October 1976

[50] SIMULA - Users Guide IBM System / 360, Norsk Regnesentral, Norwergian Computing Centre, 1971

[51] Skala V.: "Filosofie implementace relační báze dat", výzkumná zpráva 209-04-78, VŠSE Plzeň, 1978

[52] Skala V.: "Filosofie implementace relační báze dat", dílčí zpráva státního úkolu III-2-2/2

[53] Skala V.: "Použití makrogenerátoru pro editaci programů v Algolu 60 a v Algolu 68", výzkumná zpráva 209-06-78, VŠSE Plzeň, 1978

[54] Skala V.: "Implementace relační banky dat", DATABANKA 78, sborník ČSVTS dům techniky Ústí n. Labem, září 1978

[55] Smith J.M., Chang P.Y.: "Optimizing the Performance of a Relational Algebra Data Base Interface", CACM Vol. 18, No. 10, October 1975

[56] Software A.G. ADABAS: Introductory Manual

[57] Stone H.: "Introduction to Computer Organization and Data Structures", Mc Graw Hill, London 1975

[58] Sundgren B.: "An Infological Approach to Data Bases", Ph.D.Thesis, University of Stoskholm 1973

[59] Šlechta J.: "Terminologie databázových systémů", ČSVTS dům techniky Ústí n. Labem, 1975

[60] Šlechta J.: "Databázové systémy", KSNP - Praha, 1975

[61] Todd S.J.P.: "The Peterlee Relational Test Vehicle a System Overview", IBM System J., No. 4, 1976

[62] Todd S.J.P.: "Implementation of a Join Operator in Relational Data Bases", Report IBM TN 15, November 1974

[63] Tsichritzis D.: "A Network Framework for Relational Implementation", Data Base Description ed. Douque, B.C.M., Nijssen G.M., North-Holland Publishing Company, 1975

[64] Wedekind H.: "On the Selection of Access Paths in a Data Base System", Data Base Management, ed. J.W.Klimbie, K.L.Koffeman North-Holland Publishing Company 1974

[65] Wijngaarden A.: "Revised Report on the Algoritmic Language Algol 68", Springer-Verlag, New York, 1976

[66] Woodward P.M., Bound S.G.: "Algol 68-R Users Guide", Royal Radar Establishment, London, 1974

[67] Zoc I.: "Relační pohled na datové struktury", VÚMS, Aktuality výpočetní techniky 1976

13. Obsah

Přílohy:

P Ř Í L O H A A

BNF jazyka SEQUEL 2

```
<STATEMENT>::= <QUERY>
             /<DML-STATEMENT>
             /<DDL-STATEMENT>
             /<CON-STATEMENT>

<DML-STATEMENT>::= <ASSIGNMENT>
                 /<INSERTION>
                 /<DELETION>
                 /<UPDATE>

<QUERY>::= <QUERY-EXPR> ORDER BY <ORD-SPEC-LIST>
         /<QUERY-EXPR>

<ASSIGNMET>::= <RECEIVER> <- <QUERY-EXPR>

<RECEIVER>::= <TABLE-NAME>(<FIELD-NAME-LIST>)
            /<TABLE-NAME>

<INSERTION>::= INSERT INTO <RECEIVER>:<INSERT-SPEC>

<INSERT-SPEC>::= <QUERY-EXPR>
               /<LITERAL>
               /<CONSTANT>

<FIELD-NAME-LIST>::= <FIELD-NAME>
                   /<FIELD-NAME-LIST>,<FIELD-NAME>

<DELETION>::= DELETE <TABLE-NAME>
            /DELETE <TABLE-NAME><VAR-NAME>
            /DELETE <TABLE-NAME><VAR-NAME><WHERE-CLAUSE>
            /DELETE <TABLE-NAME><WHERE-CLAUSE>

<UPDATE>::= UPDATE <TABLE-NAME><VAR-NAME><SET-CLAUSE-LIST><WHERE-CLAUSE>

          /UPDATE <TABLE-NAME><VAR-NAME><SET-CLAUSE-LIST>
          /UPDATE <TABLE-NAME><SET-CLAUSE-LIST><WHERE-CLAUSE>
          /UPDATE <TABLE-NAME><SET-CLAUSE-LIST>

<WHERE-CLAUSE>::= WHERE <BOOLEAN>
                /WHERE CURRENT TUPLE OF CURSON <CURSOR-NAME>
                /WHERE CURRENT OF <CURSOR-NAME>

<SET-CLAUSE-LIST>::= <SET-CLAUSE>
                   /<SET-CLAUSE-LIST>,<SET-CLAUSE>

<SET-CLAUSE>::= SET<FIELD-NAME> = <EXPR>
              /SET<FIELD-NAME> = (<QUERY-EXPR>)

<QUERY-EXPR>::= <QUERY-BLOCK>
              /<QUERY-EXPR> <SET-OP> <QUERY-BLOCK>
              /<QUERY-EXPR>

<SET-OP>::= INTERSECT
          /UNION
```

```
61.         /MINUS
62.
63.<QUERY-BLOCK>::= <SELECT-CLAUSE> FROM <FROM-LIST><WH1><GB1>
64.               /<SELECT-CLAUSE> FROM <FROM-LIST> <WH1>
65.               /<SELECT-CLAUSE> FROM <FROM <FROM-LIST><GB1>
66.               /<SELECT-CLAUSE> FROM <FROM-LIST>
67.
68.<WH1>::=  WHERE <BOOLEAN>
69.
70.<GB1>::=  GROUP BY <FIELD-SPEC-LIST>
71.      / GROUP BY <FIELD-SPEC-LIST> HAVING <BOOLEAN>
72.
73.<SET-CLAUSE>::= SELECT <SEL-EXPR-LIST>
74.             /SELECT  *
75.
76.<SEL-EXPR-LIST>::= <SEL-EXPR>
77.                  <SEL-EXPR-LIST>,<SEL-EXPR>
78.
79.<SEL-EXPR>::= <EXPR>
80.           /<EXPR>:<HOST-LOCATION>
81.           /<VAR-NAME> . *
82.
83.<FROM-LIST>::= <TABLE-NAME>
84.            /<TABLE-NAME><VAR-NAME>
85.            /<FROM-LIST>,<TABLE-NAME>
86.            /<FROM-LIST>,<TABLE-NAME>,<VAR-NAME>
87.
88.<FIELD-SPEC-LIST>::= <FIELD-SPEC>
89.                   /<FIELD-SPEC-LIST>,<FIELD-SPEC>
90.
91.<ORD-SPEC-LIST>::= <FIELD-SPEC>
92.                 /<FIELD-SPEC><DIRECTION>
93.                 /<ORD-SPEC-LIST>,<FIELD-SPEC>
94.                 /<ORD-SPEC-LIST>,<FIELD-SPEC><DIRECTION>
95.
96.<DIRECTION>::= ASC
97.            /DESC
98.
99.<BOOLEAN>::= <BOOLEAN-TERM>
100.         /<BOOLEAN> OR <BOOLEAN-TERM>
101.
102.<BOOLEAN-TERM>::= <BOOLEAN-FACTOR>
103.                /<BOOLEAN-TERM> AND <BOOLEAN-FACTOR>
104.
105.<BOOLEAN-FACTOR>::= <BOOLEAN-PRIMARY>
106.                  /NOT<BOOLEAN-PRIMARY>
107.
108.<BOOLEAN-PRIMARY>::= <PREDICATE>
109.                   /(<BOOLEAN>)
110.
111.<PREDICATE>::= <EXPR><COMPARISON><EXPR>
112.            /<EXPR>BETWEEN<EXPR>AND<EXPR>
113.            /<EXPR><COMPARISION><TABLE-SPEC>
114.            /<FIELD-SPEC-LIST>=<FULL-TABLE-SPEC>
115.            /<FIELD-SPEC-LIST>ISIN<FULL-TABLE-SPEC>
116.            /<FIELD-SPEC-LIST>   IN<FULL-TABLE-SPEC>
117.            /IF<PREDICATE>THEN<PREDICATE>
118.            /SET(<FIELD-SPEC-LIST>)<COMPARISION><FULL-TABLE-SPEC>
119.            /SET(<FIELD-SPEC-LIST>)<COMPARISION>SET(<FIELD-SPEC-LIST>
120.            /<TABLE-SPEC><COMPARISION><FULL-TABLE-SPEC>
```

```
121.
122.<FULL-TABLE-SPEC>::= <TABLE-SPEC>
123.                    /(<ENTRY>)
124.                    /<CONSTANT>
125.
126.<TABLE-SPEC>::= <QUERY-BLOCK>
127.              /(<QUERY-EXPR>
128.              /<LITERAL>
129.
130.<EXPR>::= <ARITH-TERM>
131.       /<EXPR><ADD-OP><ARITH-TERM>
132.
133.<ARITH-TERM>::= <ARITH-FACTOR>
134.              /<ARITH-TERM><MULT-OP><ARITH-FACTOR>
135.
136.<ARITH-FACTOR>::= <ADD-OP><PRIMARY>
137.                /<PRIMARY>
138.
139.<PRIMARY>::= <FIELD-SPEC>
140.          /NEW <FIELD-SPEC>
141.          /OLD<FIELD-SPEC>
142.          /<SET-FN>(<EXPR>)
143.          /COUNT(*)
144.          /<CONSTANT>
145.          /(<EXPR>)
146.
147.<FIELD-SPEC>::= <FIELD-NAME>
148.              /<TABLE-NAME>.<FIELD-NAME>
149.              /<VAR-NAME>.<FIELD-NAME>
150.
151.<COMPARISION>::= <COMP-OP>
152.              / CONTAINS
153.              / DOES NOT CONTAIN
154.              / IN
155.              / IS IN
156.              /IS NOT IN
157.              /NOT IN
158.
159.<COMP-OP>::= =
160.          /GT
161.          /LT
162.          /GE
163.          /LE
164.          /NE
165.
166.<ADD-OP>::= +
167.         /-
168.
169.<MULT-OP>::= *
170.          /DIV
171.
172.<SET-FN>::= AVG
173.         /MAG
174.         /MIN
175.         /SUM
176.         /COUNT
177.         /<IDENTIFIER>
178.
179.<LITERAL>::= (<LIT-TUPLE-LIST>)
180.          /(<ENTRY-LIST>)
```

```
181.          /<LIT-TUPLE>
182.
183.<LIT-TUPLE-LIST>::= <LIT-TUPLE>
184.                  /<LIT-TUPLE-LIST>,<LIT-TUPLE>
185.
186.<LIT-TUPLE>::= <ENTRY>
187.             /<ENTRY-LIST>
188.
189.<ENTRY-LIST>::= <ENTRY>,<ENTRY>
190.              /<ENTRY-LIST>,<ENTRY>
191.
192.<ENTRY>::= <CONSTANT>
193.
194.<CONSTANT>::= <QUOTED-STRING>.
195.           /<NUMBER>
196.           /<HOST-LOCATION>
197.           / NULL
198.           / USER
199.           / DATE
200.           /<FIELD-NAME> OF CURSOR <CURSOR-NAME>
201.           /<FIELD-NAME>OF CURSOR <CURSOR-NAME>ON<TABLE-NAME>
202.
203.<TABLE-NAME>::= <NAME>
204.
205.<IMAGE-NAME>::= <NAME>
206.
207.<LINK-NAME>::=  <NAME>
208.
209.<ASRT-NAME>::=  <NAME>
210.
211.<TRIG-NAME>::=  <NAME>
212.
213.<NAME>::= <CREATOR>.<IDENTIFIER>
214.       /<IDENTIFIER>
215.
216.<CREATOR>::= <IDENTIFIER>
217.
218.<USER-NAME>::=<IDENTIFIER>
219.
220.<FIELD-NAME>::= <IDENTIFIER>
221.
222.<VAR-NAME>::=   <IDENTIFIER>
223.
224.<CURSOR-NAME>::=<IDENTIFIER>
225.
226.<HOST-LOCATION>::=<IDENTIFIER>
227.
228.<INTEGER>::=<NUMBER>
229.
230.<DDL-STATEMENT>::= <CREATE-TABLE>
231.                 /<EXPAND-TABLE>
232.                 /<KEEP-TABLE>
233.                 /<CREATE-IMAGE>
234.                 /<CREATE-LINK>
235.                 /<DEFINE-VIEW>
236.                 /<DROP>
237.                 /<COMMENT>
238.
239.<CREATE-TABLE>::= CREATE TABLE <TABLE-NAME>:<FIELD-DEFN-LIST>
240.
```

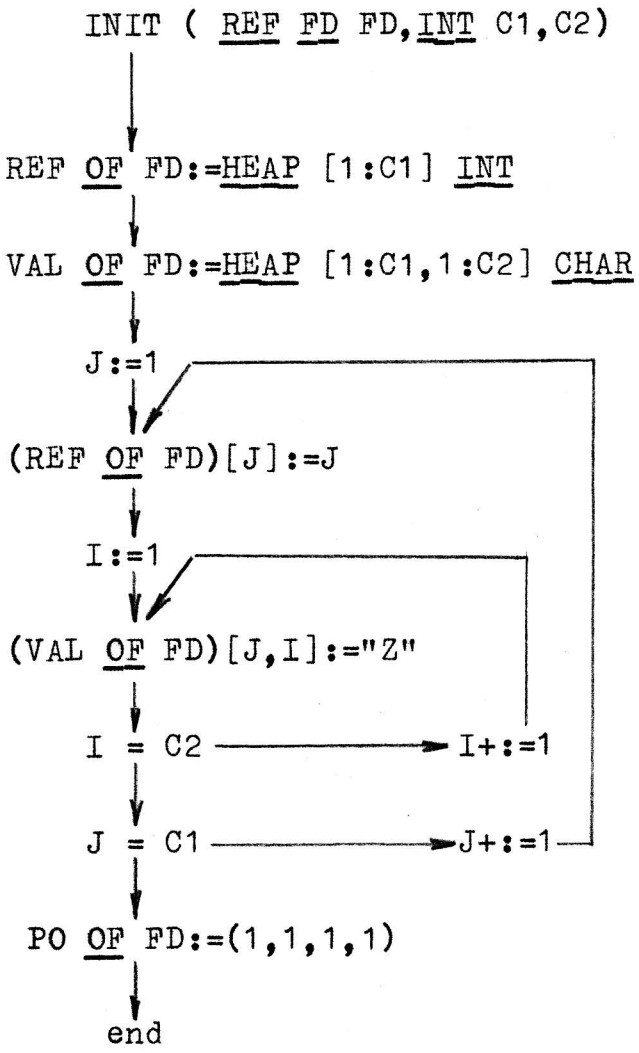
```
<FIELD-DEFN-LIST>::= <FIELD-DEFN>
                   /<FIELD-DEFN-LIST>,<FIELD-DEFN>

<FIELD-DEFN>::= <FIELD-NAME>(<TYPE>)
              /<FIELD-NAME>(<TYPE>,NONULL)

<TYPE>::= CHAR(<INTEGER>)
        /CHAR(*)
        /INTEGER
        /SMALLINT
        /DECIMAL(INTEGER,INTEGER)
        /FLOAT

<EXPAND-TABLE>::= EXPAND TABLE <TABLE-NAME> ADD FIELD <FIELD-DEFN>

<KEEP-TABLE>::= KEEP TABLE <TABLE-NAME>

<CREATE-IMAGE>::= CREATE<IMAGE-MOD-LIST>IMAGE <IMAGE-NAME1>

<IMAGE-NAME1>::= <IMAGE-NAME>ON<TABLE-NAME>(<ORD-SPEC-LIST>)

<IMAGE-MOD-LIST>::= <IMAGE-MOD>
                  /<IMAGE-MOD-LIST>,<IMAGE-MOD>

<IMAGE-MOD>::= UNIQUE
             /CLUSTERING

<CREATE-LINK>::= CREATE LINK <LINK-NAME>FROM<TABLE-NAME
                 (<FIELD-NAME-LIST>)TO <TABLE-NAME>(<FIELD-NAME-LIST>)
                 ORDERED BY<ORD-SPEC-LIST>

<DEFINE-VIEW>::= DEFINE VIEW <TABLE-NAME> AS<QUERY>
               /DEFINE VIEW <TABLE-NAME>(<FIELD-NAME-LIST>) AS <QUERY>

<DROP>::= DROP <SYSTEM-ENTITY><NAME>

<COMMENT>::= COMMENT ON <SYSTEM-ENTITY> <NAME>:<QUOTED-STRING>
           /COMMENT ON FIELD<TABLE-NAME>.<FIELD-NAME>:<QUOTED-STRING>

<SYSTEM-ENTITY>::= TABLE
                 /VIEW
                 /ASSERTION
                  /TRIGGER
                  /IMAGE
                  /LINK

<CONTROL-STATEMENT>::= <ASRT-STATEMENT>
                     /<ENFORCEMENT>
                     /<DEFINE-TRIGGER>
                     /<GRANT>
                     /<REVOKE>

<ASRT-STATEMENT>::= ASSERT<ASRT-NAME>:<BOOLEAN>.
                  /ASSERT<ASRT-NAME>IMMEDIATE:<BOOLEAN>
                  /ASSERT<ASRT-NAME>ON<ASRT-CONDITION>:<BOOLEAN>

<ASRT-CONDITION>::= <ACTION-LIST>
                  /<TABLE-NAME>
                  /<TABLE-NAME><VAR-NAME>
```

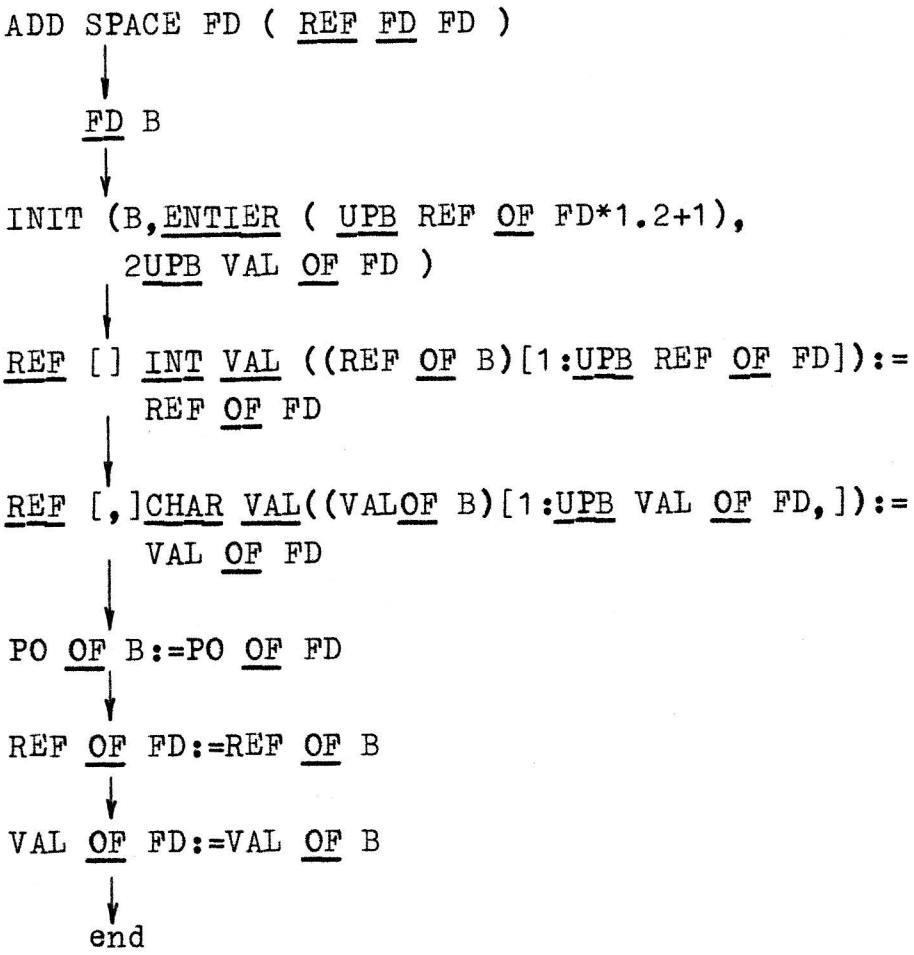
```
301.<ACTION-LIST>::= <ACTION>
302.                 <ACTION-LIST>,<ACTION>
303.
304.<ACTION>::= INSERTION OF <TABLE-NAME>
305.          /INSERTION OF <TABLE-NAME><VAR-NAME>
306.          /DELETION OF <TABLE-NAME><VAR-NAME>
307.          /DELETION OF<TABLE-NAME>
308.          /UPDATE OF<TABLE-NAME>
309.          /UPDATE OF<TABLE-NAME> <VAR-NAME>
310.          /UPDATE OF<TABLE-NAME> <VAR-NAME>.<FIELD-NAME-LIST>
311.
312.<ENFORCEMENT>::= ENFORCE INTEGRITY
313.               /ENFORCE ASSERTION <ASRT-NAME>
314.
315.<DEFINE-TRIGGER>::= DEFINE TRIGGER <TRIGGER-NAME>ON<TRIG-CONDITION>
316.                  :(STATEMENT-LIST>)
317.
318.<TRIG-CONDITION>::= <ACTION>
319.                  /READ OF<TABLE-NAME>
320.                  /READ OF<TABLE-NAME><VAR-NAME>
321.
322.<STATEMENT-LIST>::= <STATEMENT>
323.                  /<STATEMENT-LIST>;<STATEMENT>
324.
325.<GRANT>::= GRANT <AUTH><TABLE-NAME>TO<USER-LIST>
326.         GRANT <AUTH><TABLE-NAME>TO<USER-LIST> WITH GRANT OPTION
327.        /GRANT <TABLE-NAME>TO<USER>
328.        /GRANT <TABLE-NAME>TO<USER> WITH GRANT OPTION
329.
330.<AUTH>::= ALL RIGHTS ON
331.       /<OPERATION-LIST> ON
332.       /ALL BUT <OPERATION-LIST> ON
333.
334.<USER-LIST>::= <USER-NAME>
335.             /<USER-LIST>,<USER-NAME>
336.             / PUBLIC
337.
338.<OPERATION-LIST>::= <OPERATION>
339.                   /<OPERATION-LIST>,<OPERATION>
340.
341.<OPERATION>::= READ
342.             /INSERT
343.             /DELETE
344.             /UPDATE(<FIELD-NAME-LIST>)
345.             /UPDATE
346.             /DROP
347.             /EXPAND
348.             /IMAGE
349.             /LINK
350.             /CONTROL
351.
352.<REVOKE>::= REVOKE <TABLE-NAME> FROM <USER-LIST>
353.                /<OPERATION-LIST>ON<TABLE-NAME>FROM<USER-LIST>
354.
355.
356./*
357.
```

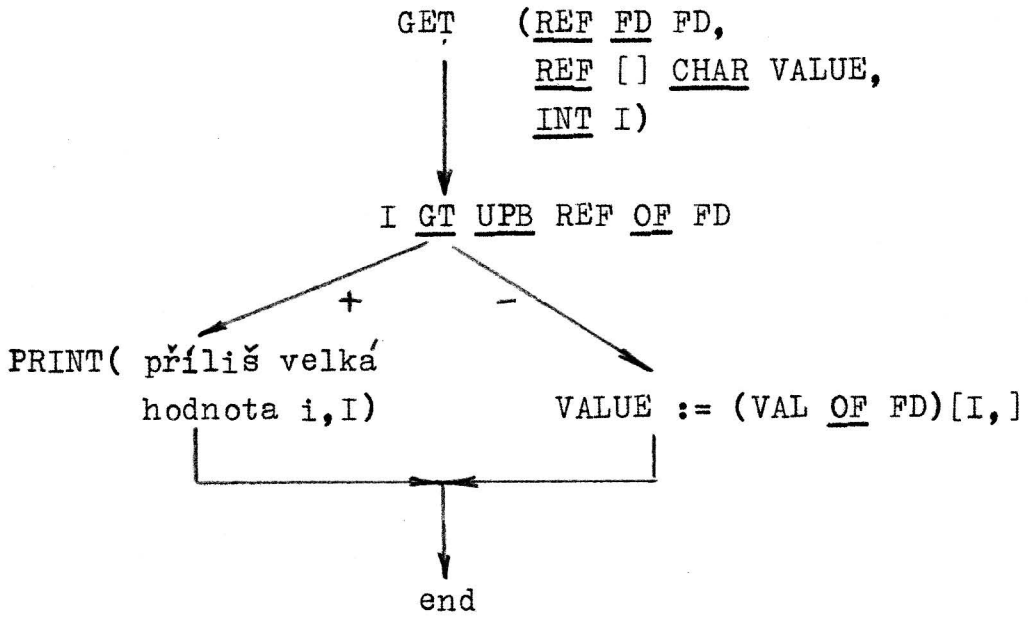
```
-----  TIME CONSUMED BY MONITOR          24  -----
-----  TIME CONSUMED BY /MACRO         1:24  -----
-----  TIME CONSUMED BY MONITOR           1  -----
-----  TIME CONSUMED BY JOB            1:49  -----
```

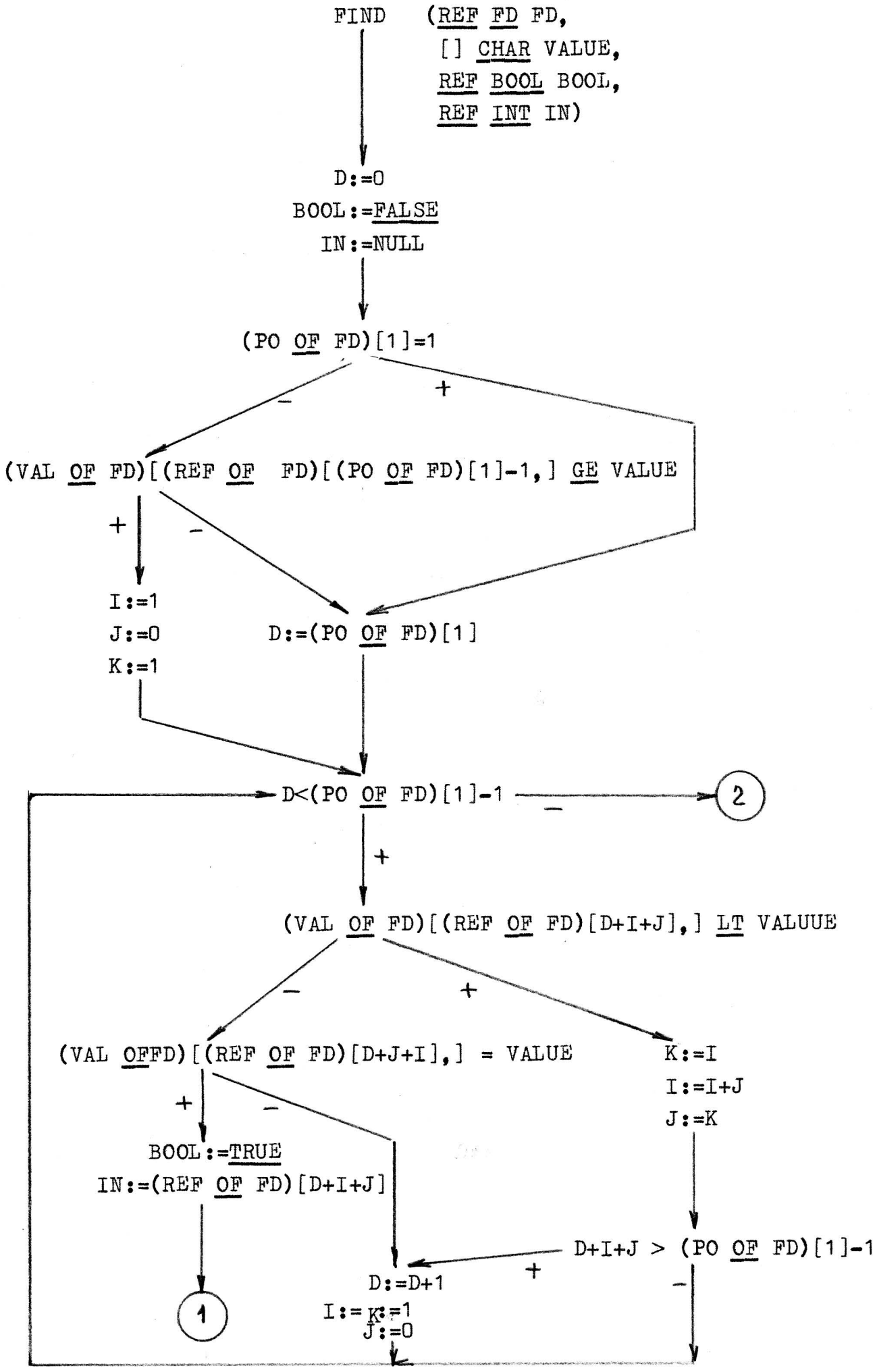
P Ř Í L O H A B

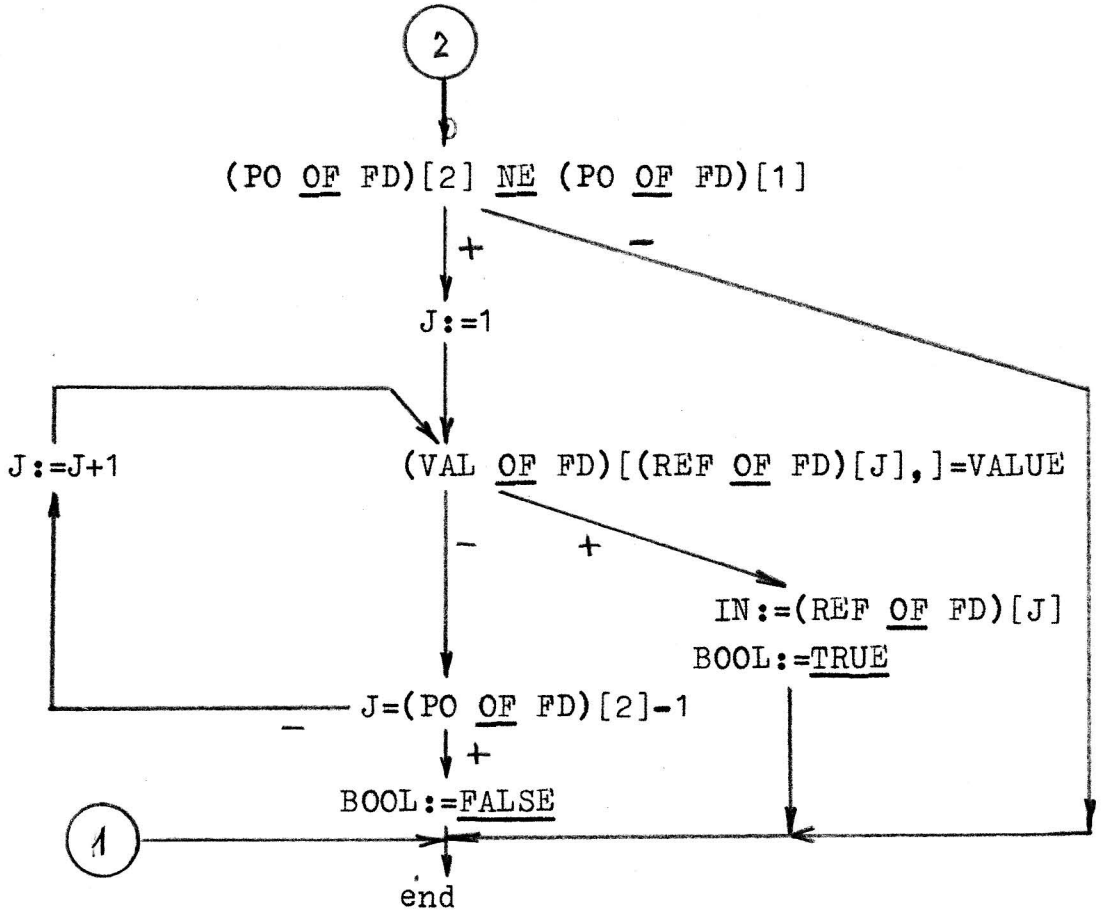
Podrobné vývojové diagramy operací nad fd

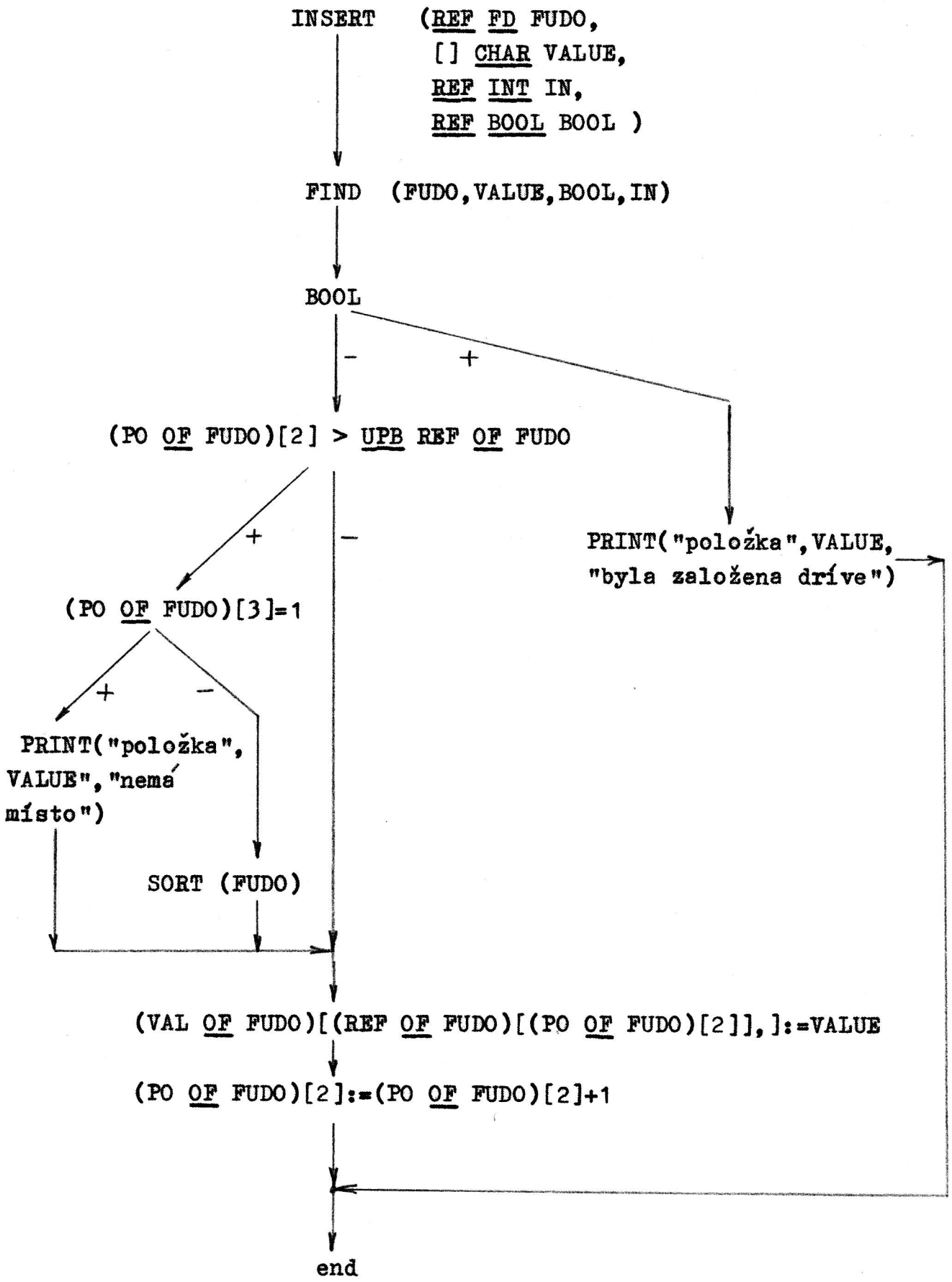

INIT (REF FD FD,INT C1,C2)
REF OF FD:=HEAP [1:C1] INT
VAL OF FD:=HEAP [1:C1,1:C2] CHAR
J:=1
(REF OF FD)[J]:=J
I:=1
(VAL OF FD)[J,I]:="Z"
I = C2
I+:=1
J = C1
J+:=1
PO OF FD:=(1,1,1,1)
end

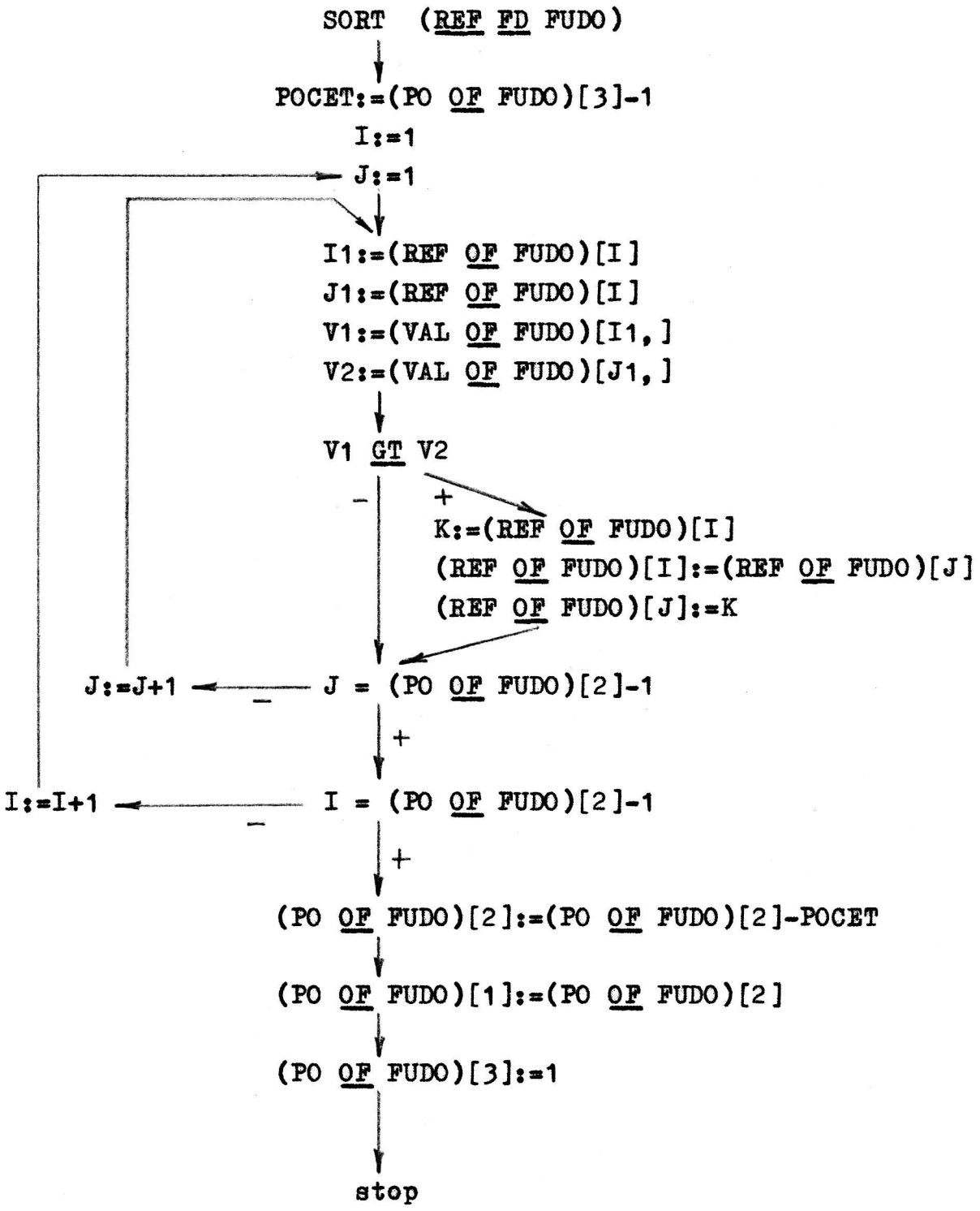

ADD SPACE FD (REF FD FD)
FD B
INIT (B,ENTIER (UPB REF OF FD*1.2+1),
2UPB VAL OF FD)
REF [] INT VAL ((REF OF B)[1:UPB REF OF FD]):=
REF OF FD
REF [,]CHAR VAL((VALOF B)[1:UPB VAL OF FD,]):=
VAL OF FD
PO OF B:=PO OF FD
REF OF FD:=REF OF B
VAL OF FD:=VAL OF B
end

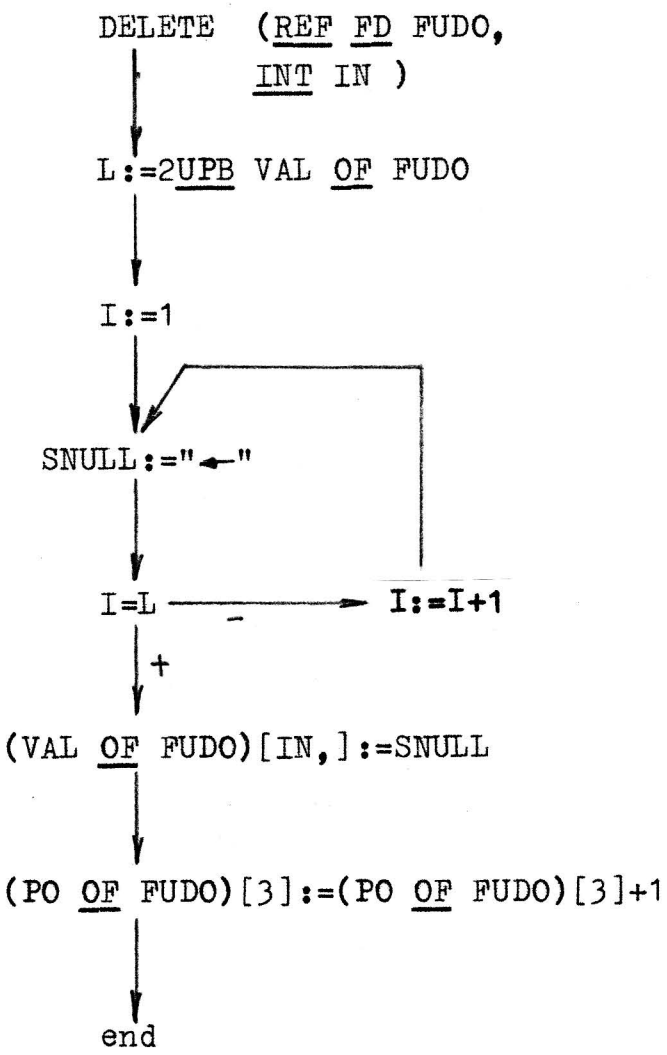

GET (REF FD FD,
REF [] CHAR VALUE,
INT I)
I GT UPB REF OF FD
+
-
PRINT(příliš velká
hodnota i,I)
VALUE := (VAL OF FD)[I,]
end

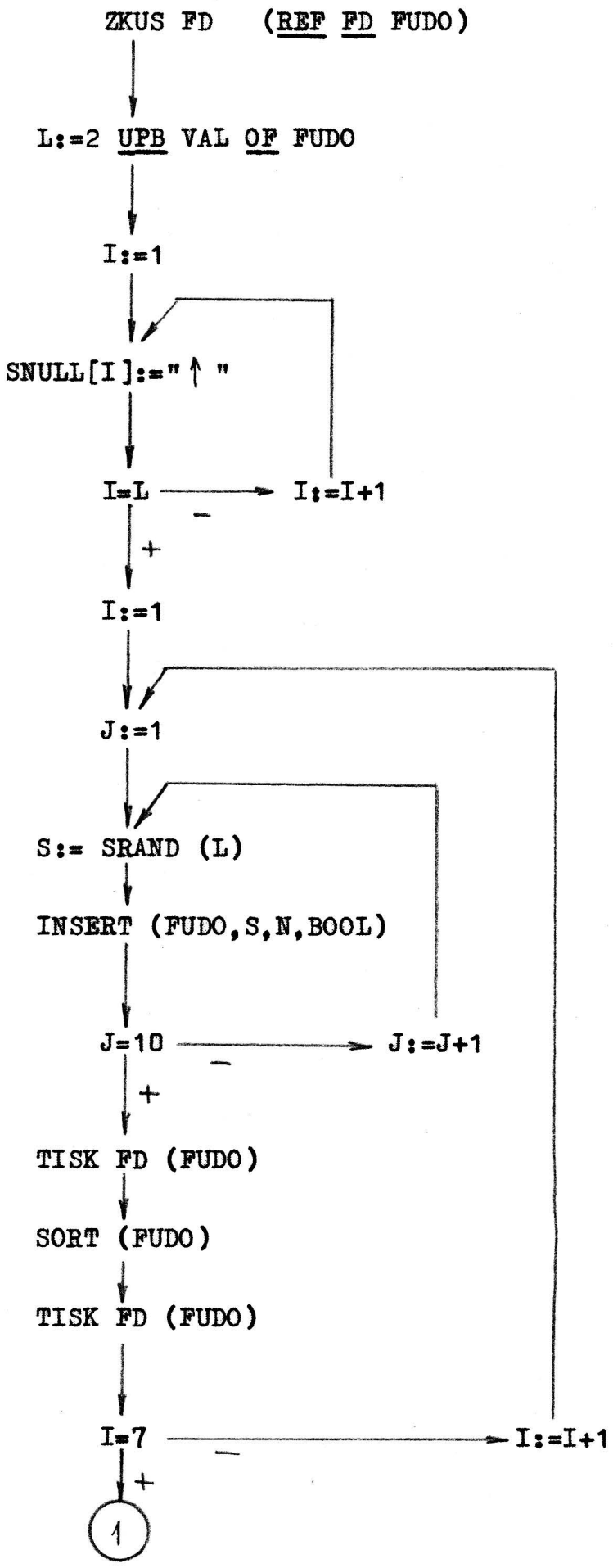

FIND (REF FD FD,
[] CHAR VALUE,
REF BOOL BOOL,
REF INT IN)
D:=0
BOOL:=FALSE
IN:=NULL
(PO OF FD)[1]=1
−
+
(VAL OF FD)[(REF OF FD)[(PO OF FD)[1]-1,] GE VALUE
+
−
I:=1
J:=0
K:=1
D:=(PO OF FD)[1]
D<(PO OF FD)[1]-1
−
2
+
(VAL OF FD)[(REF OF FD)[D+I+J],] LT VALUUE
−
+
(VAL OFFD)[(REF OF FD)[D+J+I],] = VALUE
K:=I
I:=I+J
J:=K
+
−
BOOL:=TRUE
IN:=(REF OF FD)[D+I+J]
D+I+J > (PO OF FD)[1]-1
+
−
D:=D+1
1
I:=K:=1
J:=0

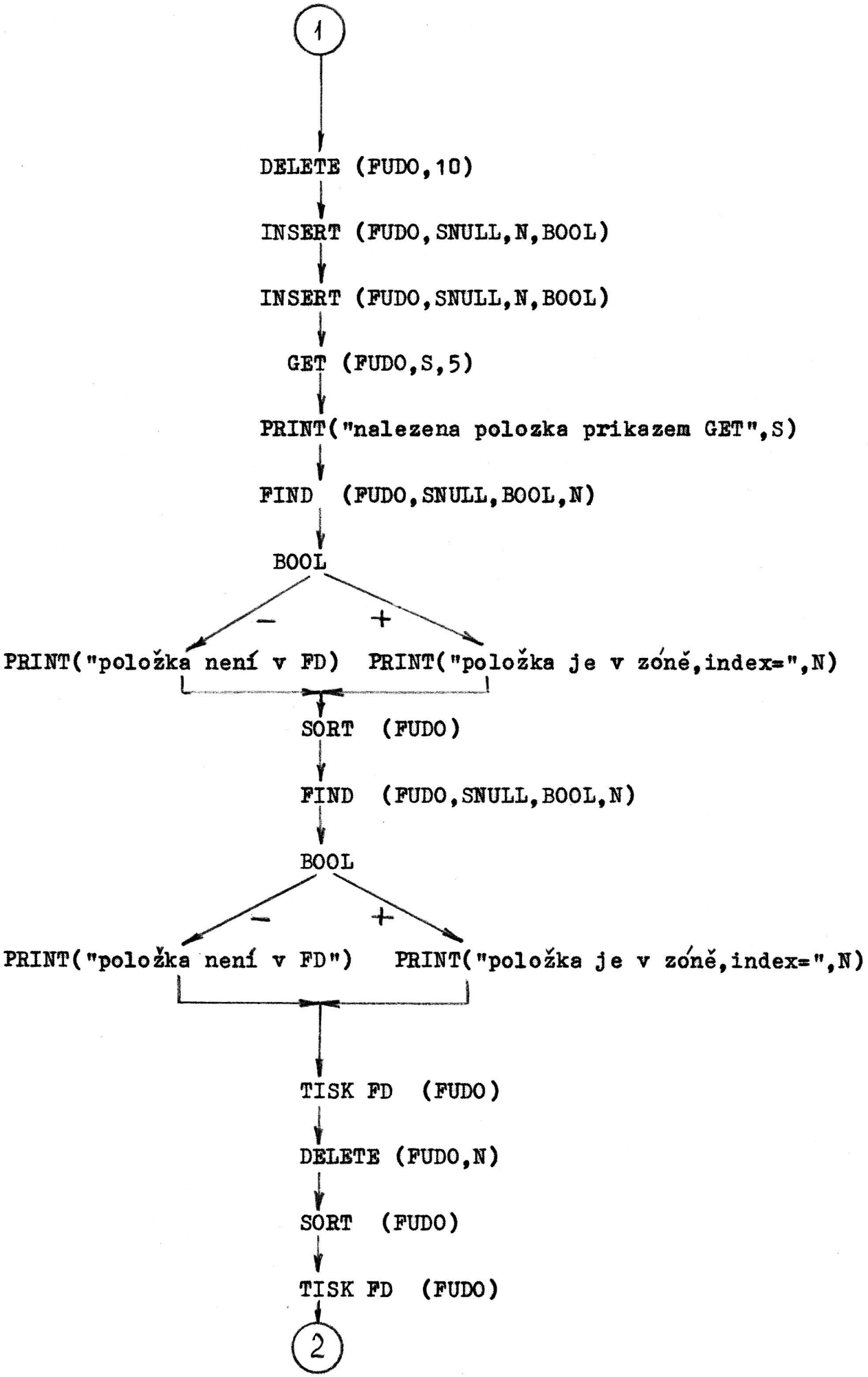

2
(PO OF FD)[2] NE (PO OF FD)[1]
+
-
J:=1
J:=J+1
(VAL OF FD)[(REF OF FD)[J],]=VALUE
-
+
IN:=(REF OF FD)[J]
BOOL:=TRUE
J=(PO OF FD)[2]-1
-
+
BOOL:=FALSE
1
end

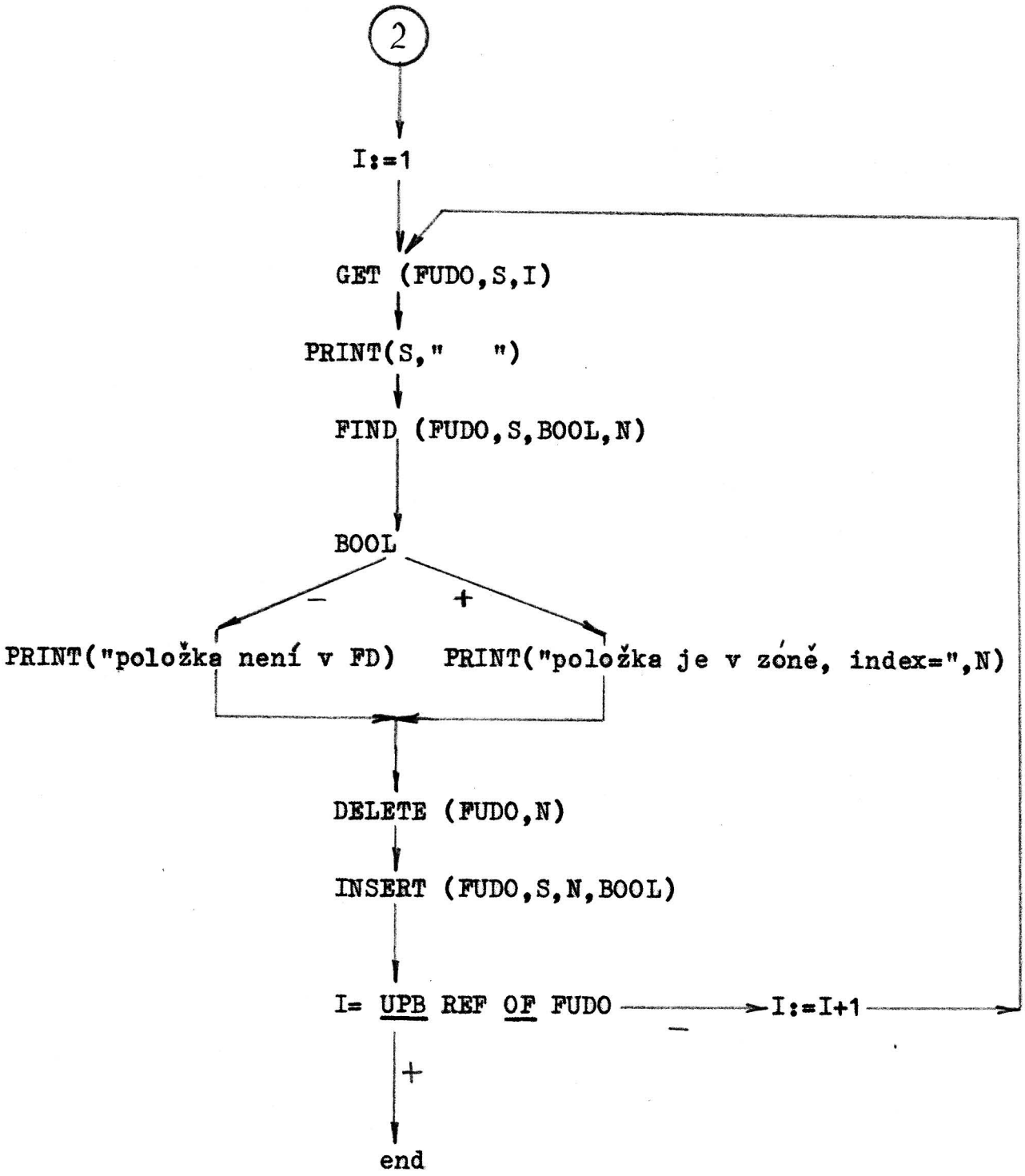

INSERT (REF FD FUDO,
[] CHAR VALUE,
REF INT IN,
REF BOOL BOOL)
FIND (FUDO,VALUE,BOOL,IN)
BOOL
-
+
(PO OF FUDO)[2] > UPB REF OF FUDO
+
-
PRINT("položka",VALUE,
"byla založena dríve")
(PO OF FUDO)[3]=1
+
-
PRINT("položka",
VALUE","nemá
místo")
SORT (FUDO)
(VAL OF FUDO)[(REF OF FUDO)[(PO OF FUDO)[2]],]:=VALUE
(PO OF FUDO)[2]:=(PO OF FUDO)[2]+1
end


SORT (REF FD FUDO)
POCET:=(PO OF FUDO)[3]-1
I:=1
J:=1
I1:=(REF OF FUDO)[I]
J1:=(REF OF FUDO)[I]
V1:=(VAL OF FUDO)[I1,]
V2:=(VAL OF FUDO)[J1,]
V1 GT V2
-
+
K:=(REF OF FUDO)[I]
(REF OF FUDO)[I]:=(REF OF FUDO)[J]
(REF OF FUDO)[J]:=K
J:=J+1
J = (PO OF FUDO)[2]-1
-
+
I:=I+1
I = (PO OF FUDO)[2]-1
-
+
(PO OF FUDO)[2]:=(PO OF FUDO)[2]-POCET
(PO OF FUDO)[1]:=(PO OF FUDO)[2]
(PO OF FUDO)[3]:=1
stop


DELETE (REF FD FUDO,
INT IN)
L:=2UPB VAL OF FUDO
I:=1
SNULL:="←"
I=L
I:=I+1
-
+
(VAL OF FUDO)[IN,]:=SNULL
(PO OF FUDO)[3]:=(PO OF FUDO)[3]+1
end

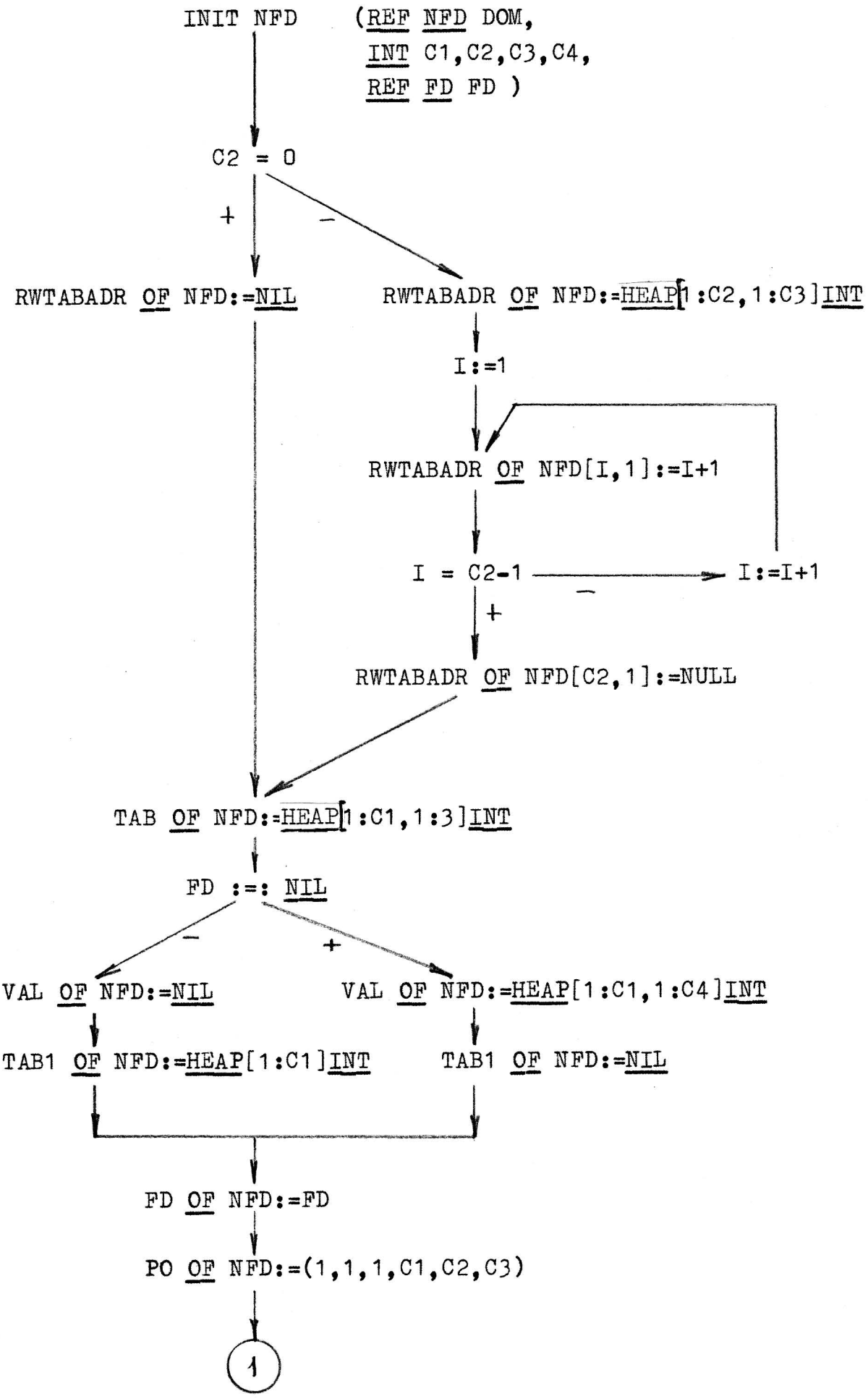

ZKUS FD (REF FD FUDO)
L:=2 UPB VAL OF FUDO
I:=1
SNULL[I]:="↑ "
I=L
I:=I+1
-
+
I:=1
J:=1
S:= SRAND (L)
INSERT (FUDO,S,N,BOOL)
J=10
J:=J+1
-
+
TISK FD (FUDO)
SORT (FUDO)
TISK FD (FUDO)
I=7
I:=I+1
-
+
1

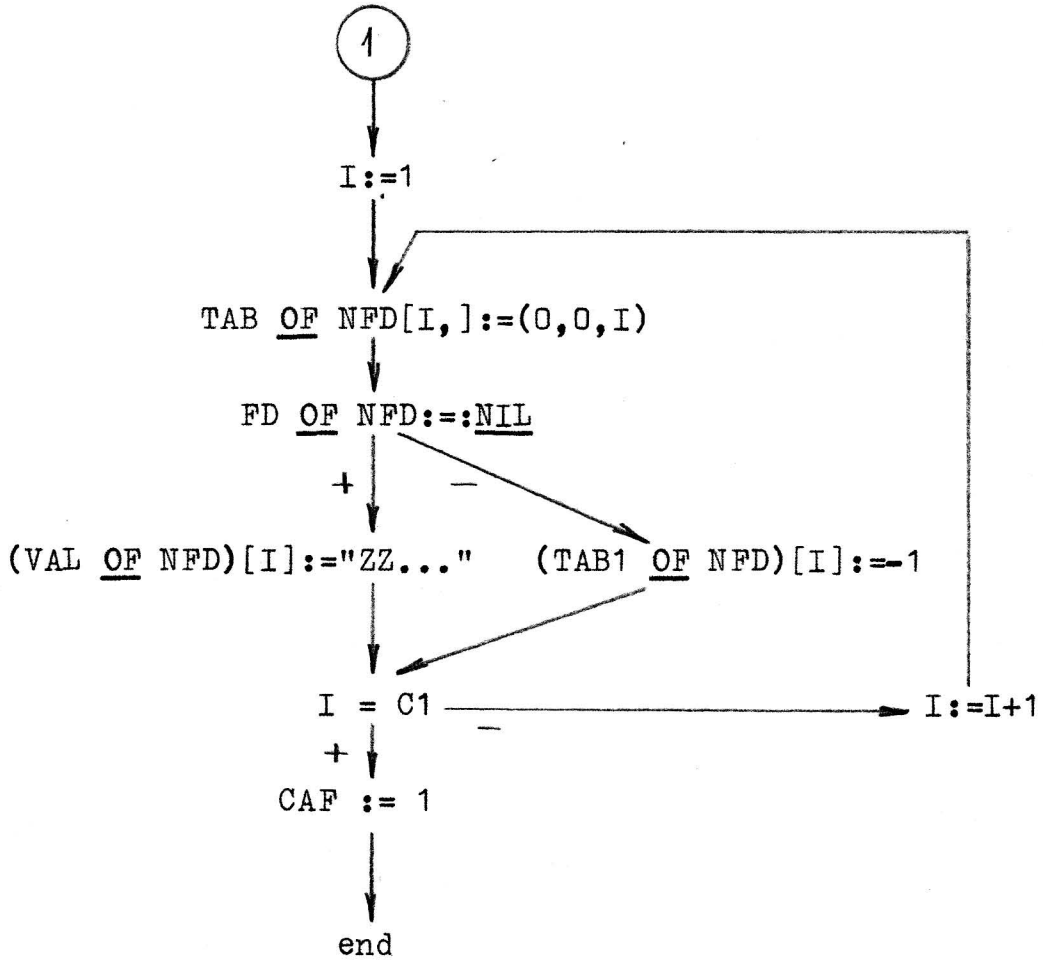

1
DELETE (FUDO,10)
INSERT (FUDO,SNULL,N,BOOL)
INSERT (FUDO,SNULL,N,BOOL)
GET (FUDO,S,5)
PRINT("nalezena polozka prikazem GET",S)
FIND (FUDO,SNULL,BOOL,N)
BOOL
-
+
PRINT("položka není v FD)
PRINT("položka je v zóně,index=",N)
SORT (FUDO)
FIND (FUDO,SNULL,BOOL,N)
BOOL
-
+
PRINT("položka není v FD")
PRINT("položka je v zóně,index=",N)
TISK FD (FUDO)
DELETE (FUDO,N)
SORT (FUDO)
TISK FD (FUDO)
2

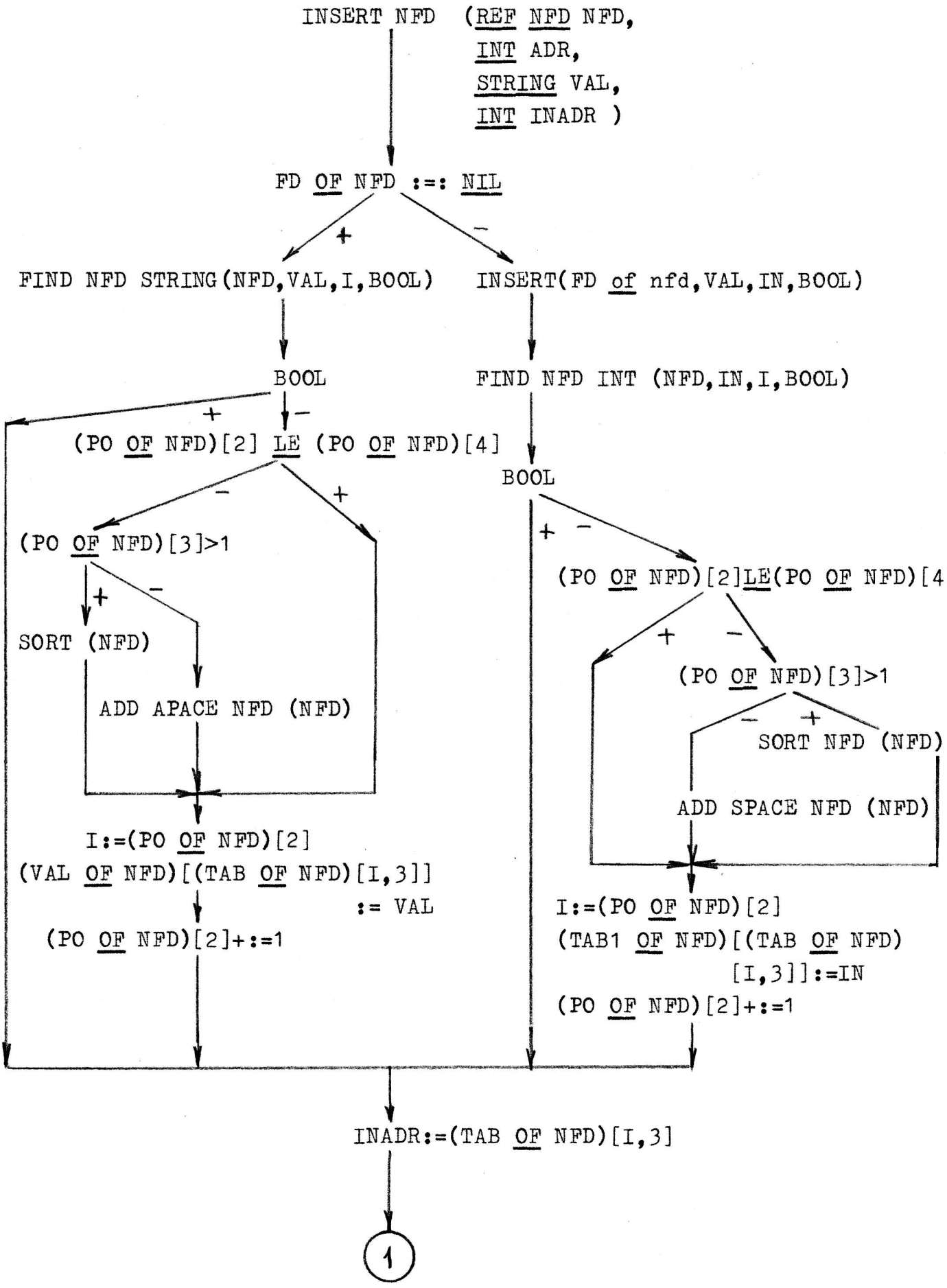

2
I:=1
GET (FUDO,S,I)
PRINT(S," ")
FIND (FUDO,S,BOOL,N)
BOOL
-
+
PRINT("položka není v FD)
PRINT("položka je v zóně, index=",N)
DELETE (FUDO,N)
INSERT (FUDO,S,N,BOOL)
I= UPB REF OF FUDO
-
I:=I+1
+
end

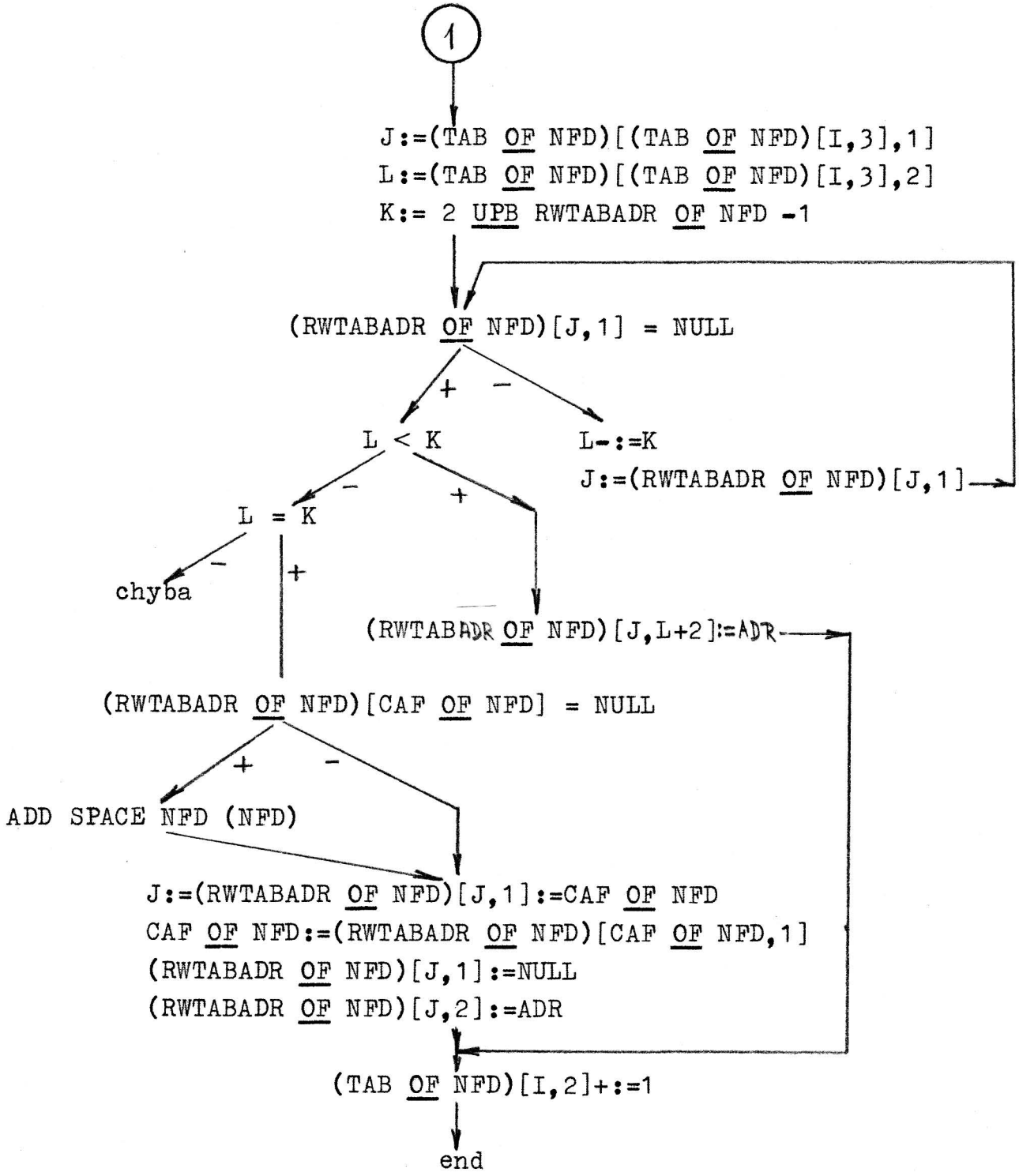
P Ř Í L O H A C

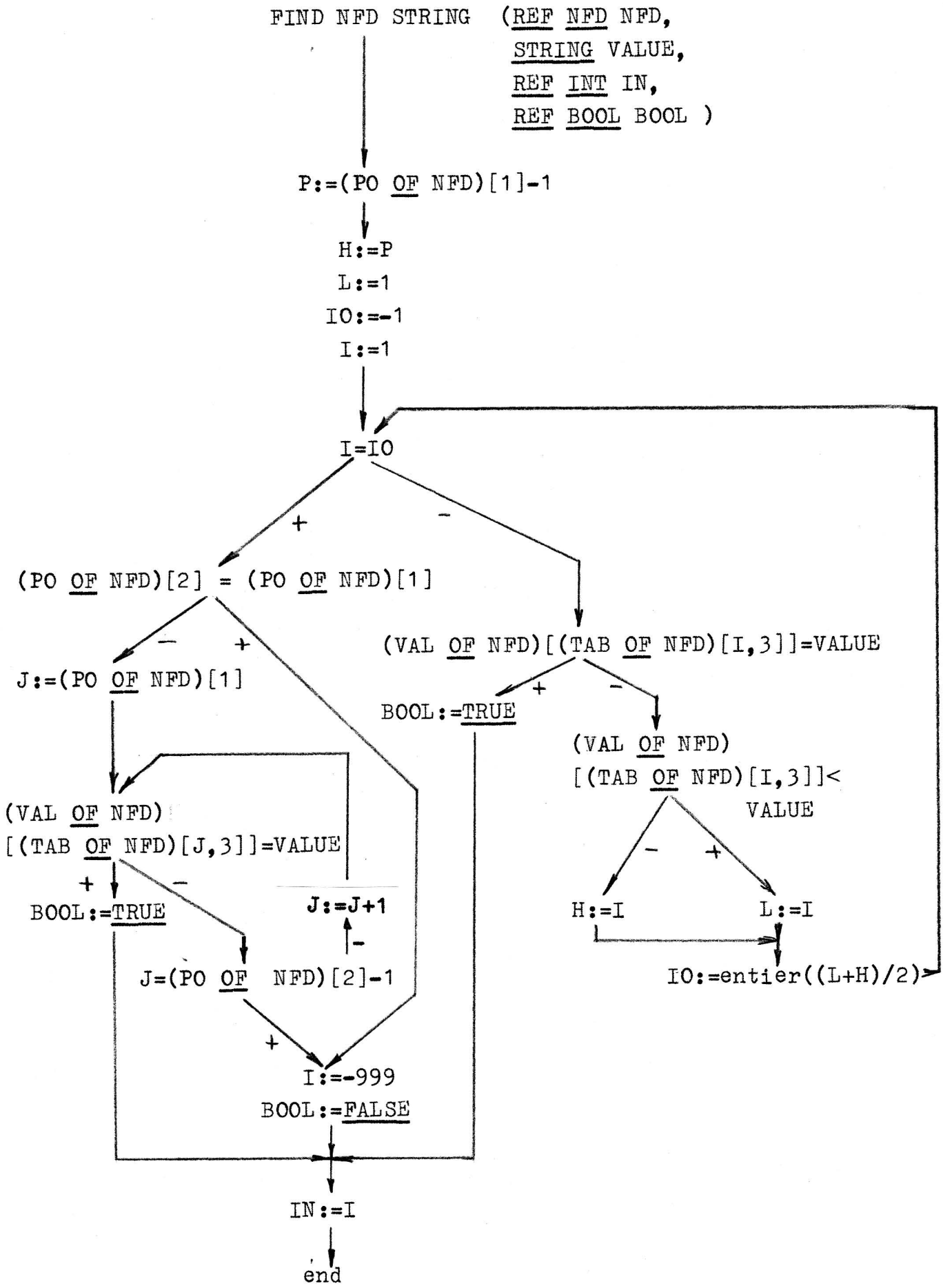
Podrobné vývojové diagramy operací nad nfd

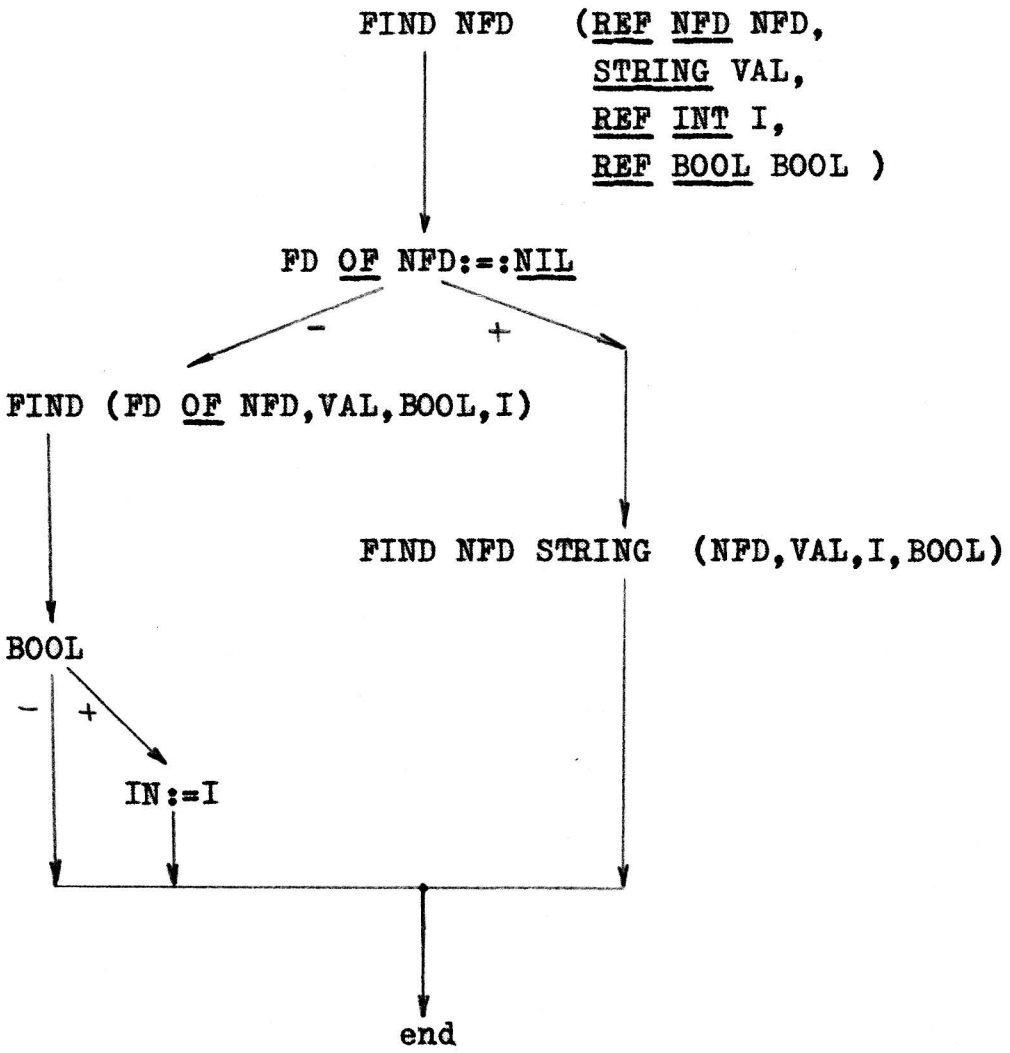

INIT NFD
(REF NFD DOM,
INT C1,C2,C3,C4,
REF FD FD)
C2 = 0
+
−
RWTABADR OF NFD:=NIL
RWTABADR OF NFD:=HEAP[1:C2,1:C3]INT
I:=1
RWTABADR OF NFD[I,1]:=I+1
I = C2-1
I:=I+1
−
+
RWTABADR OF NFD[C2,1]:=NULL
TAB OF NFD:=HEAP[1:C1,1:3]INT
FD :=: NIL
−
+
VAL OF NFD:=NIL
VAL OF NFD:=HEAP[1:C1,1:C4]INT
TAB1 OF NFD:=HEAP[1:C1]INT
TAB1 OF NFD:=NIL
FD OF NFD:=FD
PO OF NFD:=(1,1,1,C1,C2,C3)
1

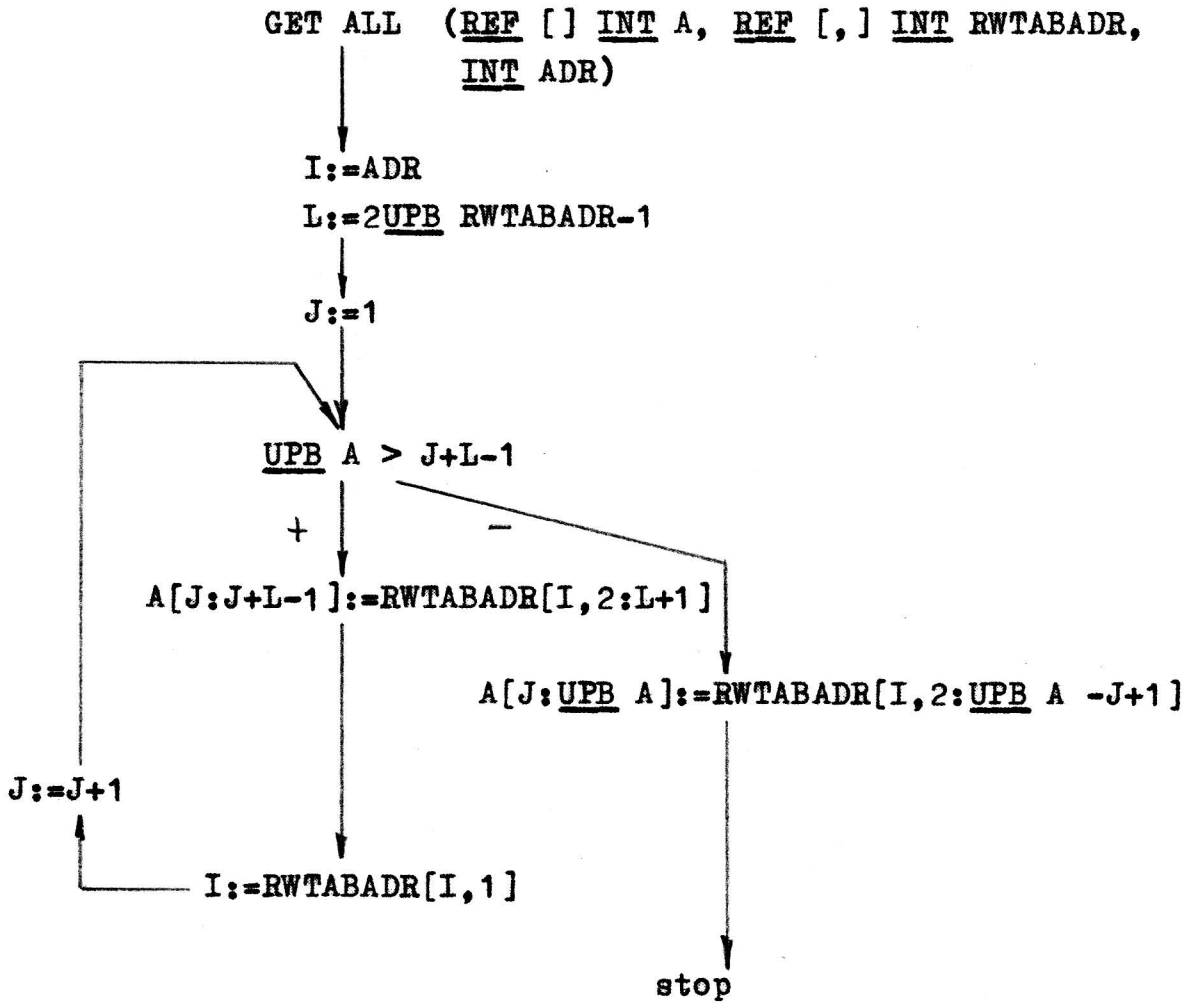

1
I:=1
TAB OF NFD[I,]:=(0,0,I)
FD OF NFD:=:NIL
+
—
(VAL OF NFD)[I]:="ZZ..."
(TAB1 OF NFD)[I]:=-1
I = C1
I:=I+1
—
+
CAF := 1
end

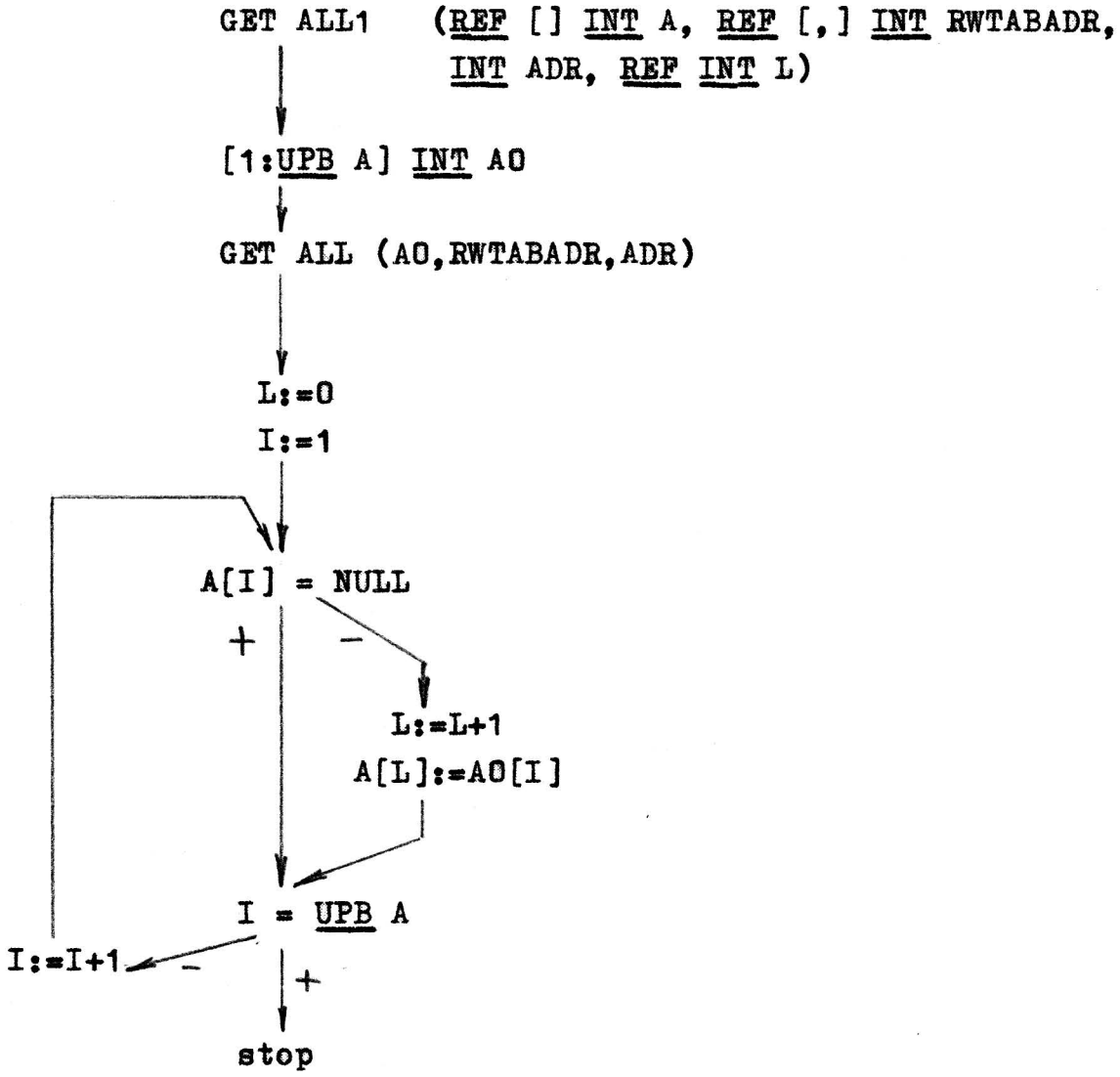

INSERT NFD (REF NFD NFD,
INT ADR,
STRING VAL,
INT INADR)
FD OF NFD :=: NIL
+
-
FIND NFD STRING(NFD,VAL,I,BOOL)
INSERT(FD of nfd,VAL,IN,BOOL)
BOOL
FIND NFD INT (NFD,IN,I,BOOL)
+
-
(PO OF NFD)[2] LE (PO OF NFD)[4]
BOOL
-
+
(PO OF NFD)[3]>1
+ -
(PO OF NFD)[2]LE(PO OF NFD)[4]
+
-
SORT (NFD)
ADD APACE NFD (NFD)
+ -
(PO OF NFD)[3]>1
- +
SORT NFD (NFD)
ADD SPACE NFD (NFD)
I:=(PO OF NFD)[2]
(VAL OF NFD)[(TAB OF NFD)[I,3]]
:= VAL
(PO OF NFD)[2]+:=1
I:=(PO OF NFD)[2]
(TAB1 OF NFD)[(TAB OF NFD)
[I,3]]:=IN
(PO OF NFD)[2]+:=1
INADR:=(TAB OF NFD)[I,3]
1

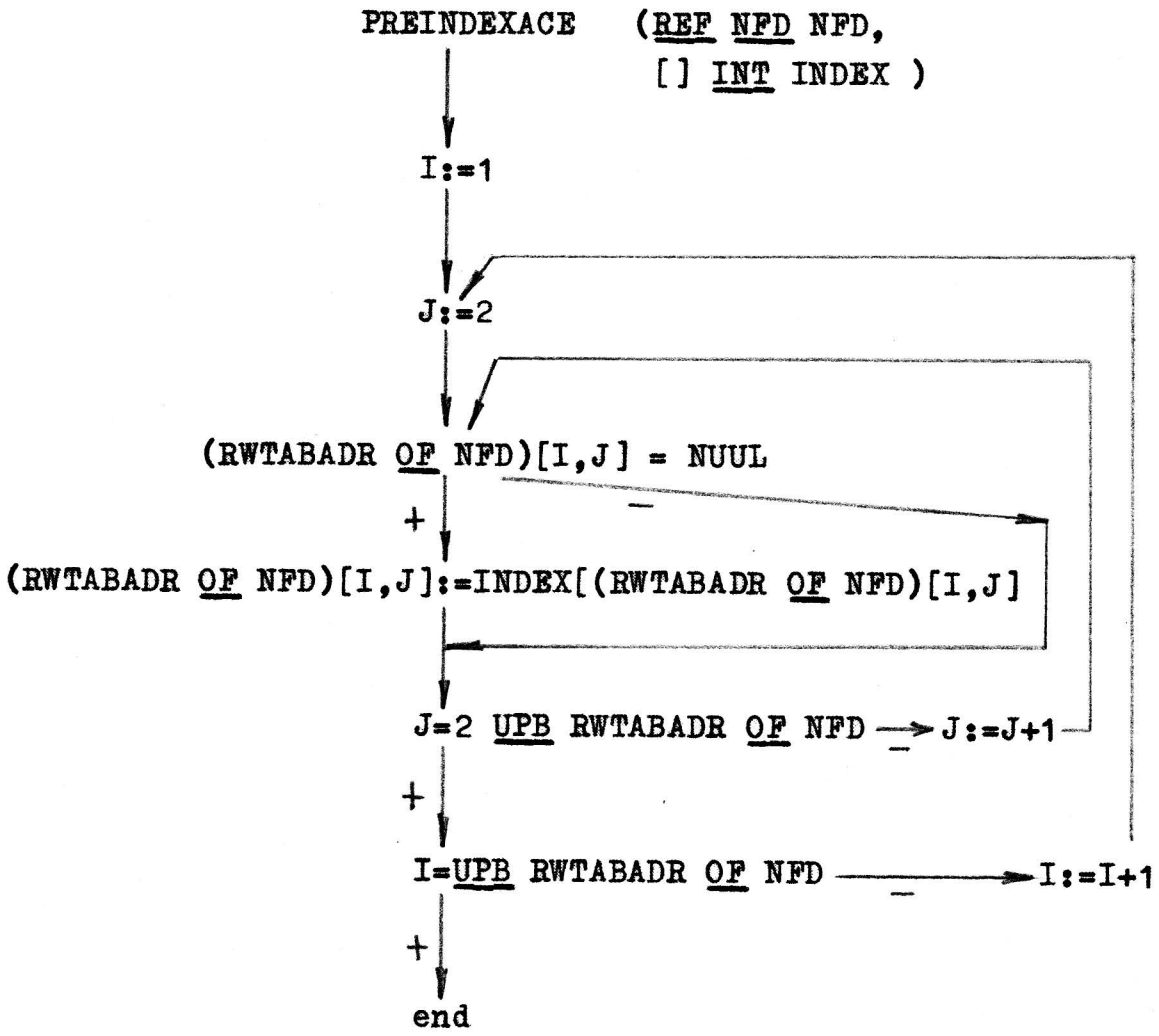

1
J:=(TAB OF NFD)[(TAB OF NFD)[I,3],1]
L:=(TAB OF NFD)[(TAB OF NFD)[I,3],2]
K:= 2 UPB RWTABADR OF NFD -1
(RWTABADR OF NFD)[J,1] = NULL
+
-
L < K
L-:=K
J:=(RWTABADR OF NFD)[J,1]
-
+
L = K
-
+
chyba
(RWTABADR OF NFD)[J,L+2]:=ADR
(RWTABADR OF NFD)[CAF OF NFD] = NULL
+
-
ADD SPACE NFD (NFD)
J:=(RWTABADR OF NFD)[J,1]:=CAF OF NFD
CAF OF NFD:=(RWTABADR OF NFD)[CAF OF NFD,1]
(RWTABADR OF NFD)[J,1]:=NULL
(RWTABADR OF NFD)[J,2]:=ADR
(TAB OF NFD)[I,2]+:=1
end

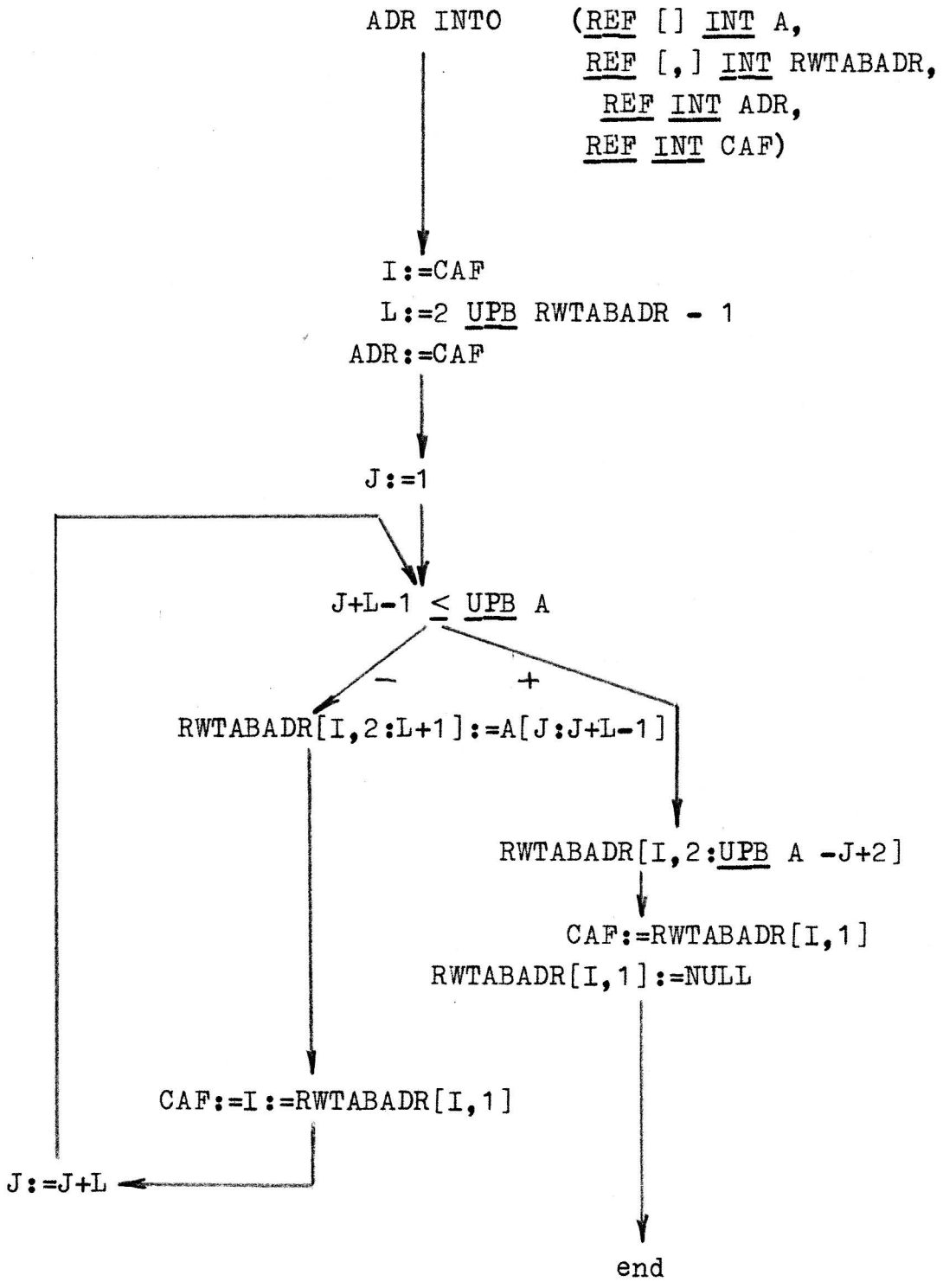

FIND NFD STRING (REF NFD NFD,
STRING VALUE,
REF INT IN,
REF BOOL BOOL)
P:=(PO OF NFD)[1]-1
H:=P
L:=1
IO:=-1
I:=1
I=IO
+
-
(PO OF NFD)[2] = (PO OF NFD)[1]
(VAL OF NFD)[(TAB OF NFD)[I,3]]=VALUE
-
+
J:=(PO OF NFD)[1]
+
-
BOOL:=TRUE
(VAL OF NFD)
[(TAB OF NFD)[I,3]]<
VALUE
(VAL OF NFD)
[(TAB OF NFD)[J,3]]=VALUE
+
-
BOOL:=TRUE
J:=J+1
-
H:=I
L:=I
J=(PO OF NFD)[2]-1
IO:=entier((L+H)/2)
+
I:=-999
BOOL:=FALSE
IN:=I
end

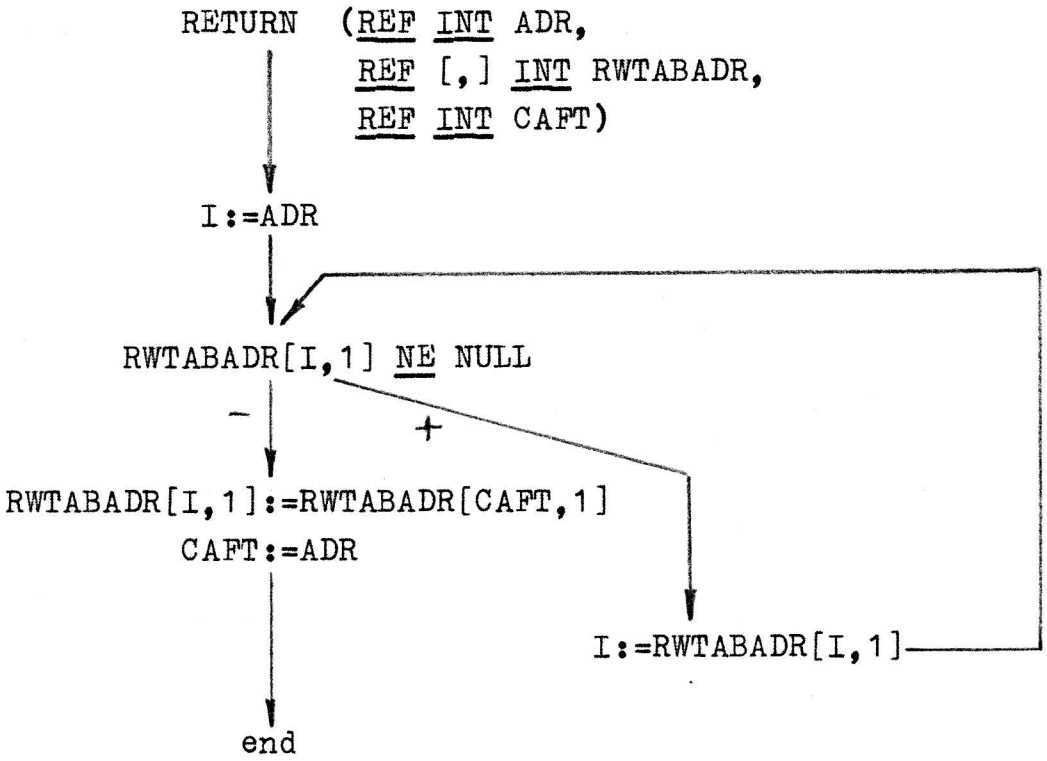

FIND NFD (REF NFD NFD,
STRING VAL,
REF INT I,
REF BOOL BOOL)
FD OF NFD:=:NIL
-
+
FIND (FD OF NFD,VAL,BOOL,I)
FIND NFD STRING (NFD,VAL,I,BOOL)
BOOL
-
+
IN:=I
end

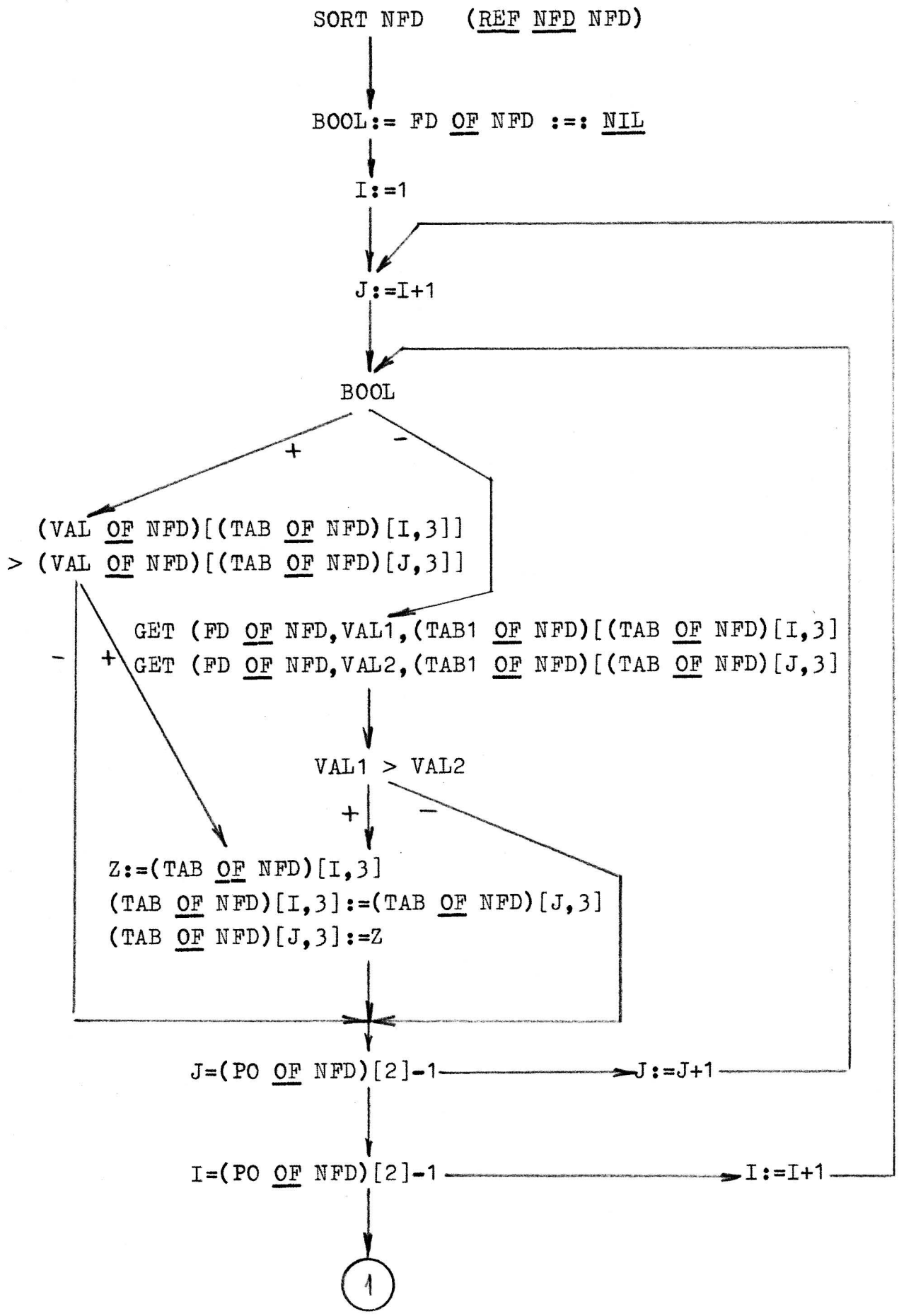

GET ALL (REF [] INT A, REF [,] INT RWTABADR,
INT ADR)
I:=ADR
L:=2UPB RWTABADR-1
J:=1
UPB A > J+L-1
+
−
A[J:J+L-1]:=RWTABADR[I,2:L+1]
A[J:UPB A]:=RWTABADR[I,2:UPB A -J+1]
J:=J+1
I:=RWTABADR[I,1]
stop

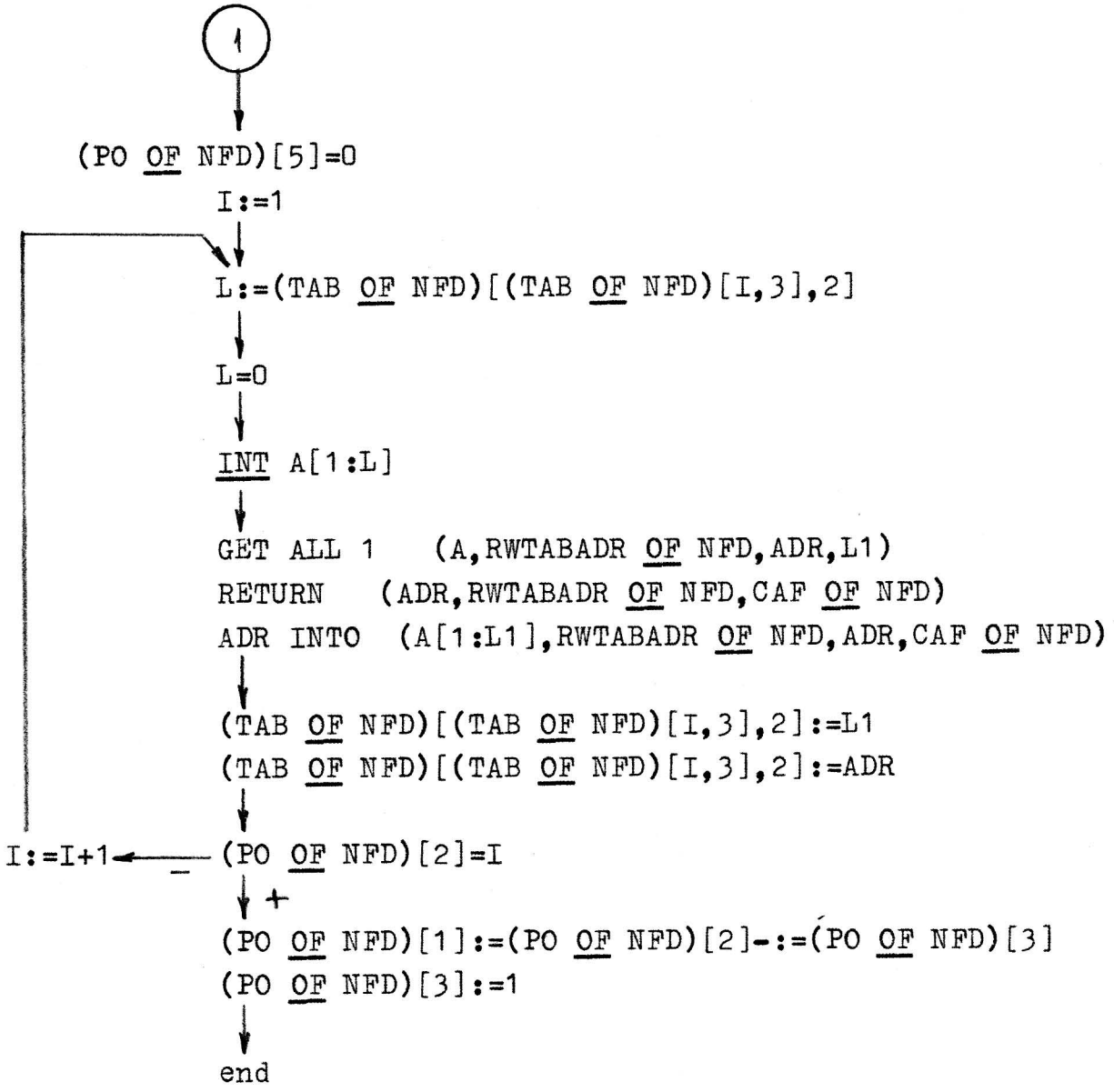

GET ALL1 (REF [] INT A, REF [,] INT RWTABADR,
INT ADR, REF INT L)
[1:UPB A] INT AO
GET ALL (AO,RWTABADR,ADR)
L:=0
I:=1
A[I] = NULL
+
-
L:=L+1
A[L]:=AO[I]
I = UPB A
I:=I+1
-
+
stop


PREINDEXACE (REF NFD NFD,
[] INT INDEX)
I:=1
J:=2
(RWTABADR OF NFD)[I,J] = NUUL
+
(RWTABADR OF NFD)[I,J]:=INDEX[(RWTABADR OF NFD)[I,J]
J=2 UPB RWTABADR OF NFD
J:=J+1
+
I=UPB RWTABADR OF NFD
I:=I+1
+
end


ADR INTO (REF [] INT A,
REF [,] INT RWTABADR,
REF INT ADR,
REF INT CAF)
I:=CAF
L:=2 UPB RWTABADR - 1
ADR:=CAF
J:=1
J+L-1 ≤ UPB A
−
+
RWTABADR[I,2:L+1]:=A[J:J+L-1]
RWTABADR[I,2:UPB A -J+2]
CAF:=RWTABADR[I,1]
RWTABADR[I,1]:=NULL
CAF:=I:=RWTABADR[I,1]
J:=J+L
end

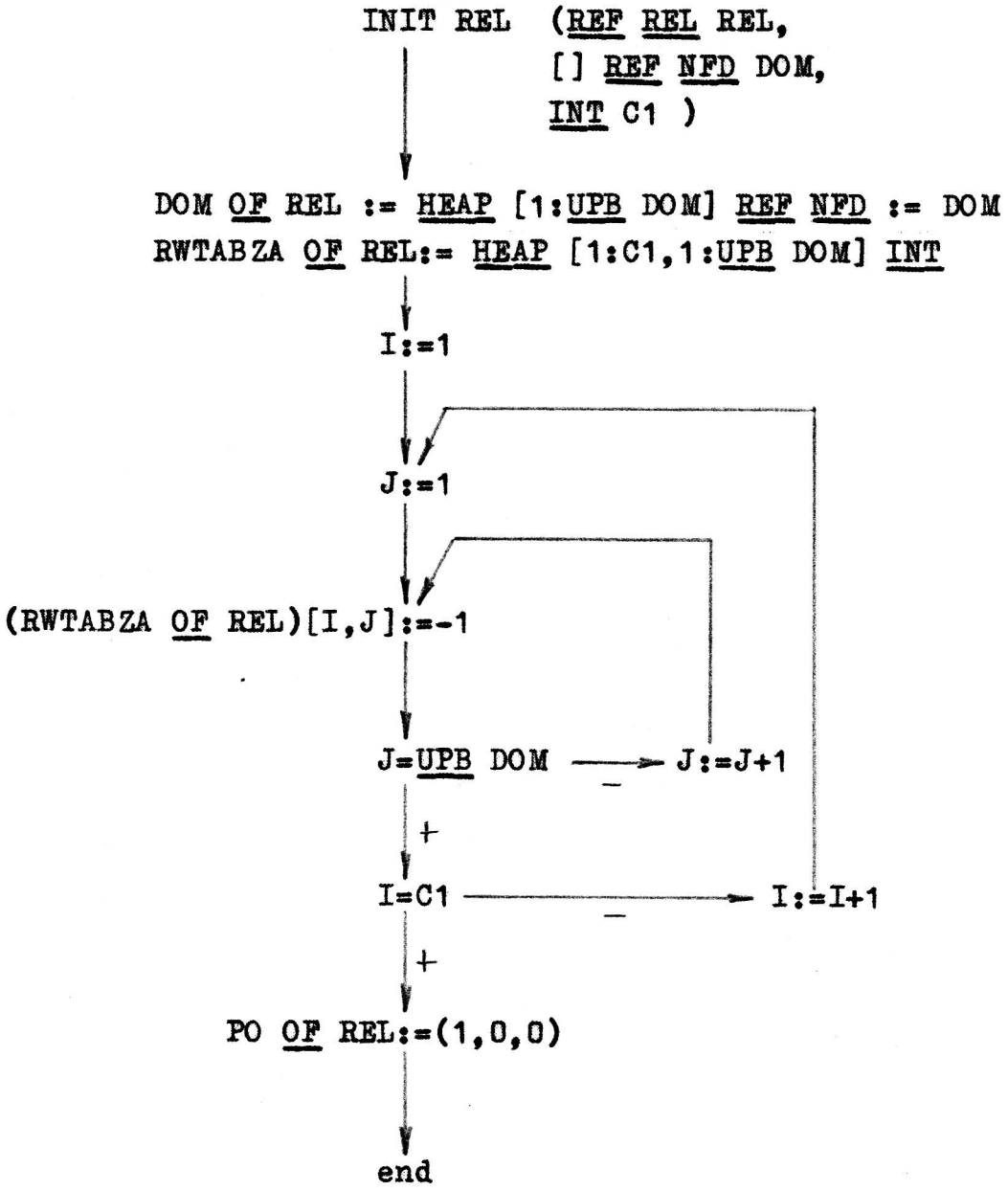

RETURN (REF INT ADR,
REF [,] INT RWTABADR,
REF INT CAFT)
I:=ADR
RWTABADR[I,1] NE NULL
-
+
RWTABADR[I,1]:=RWTABADR[CAFT,1]
CAFT:=ADR
I:=RWTABADR[I,1]
end

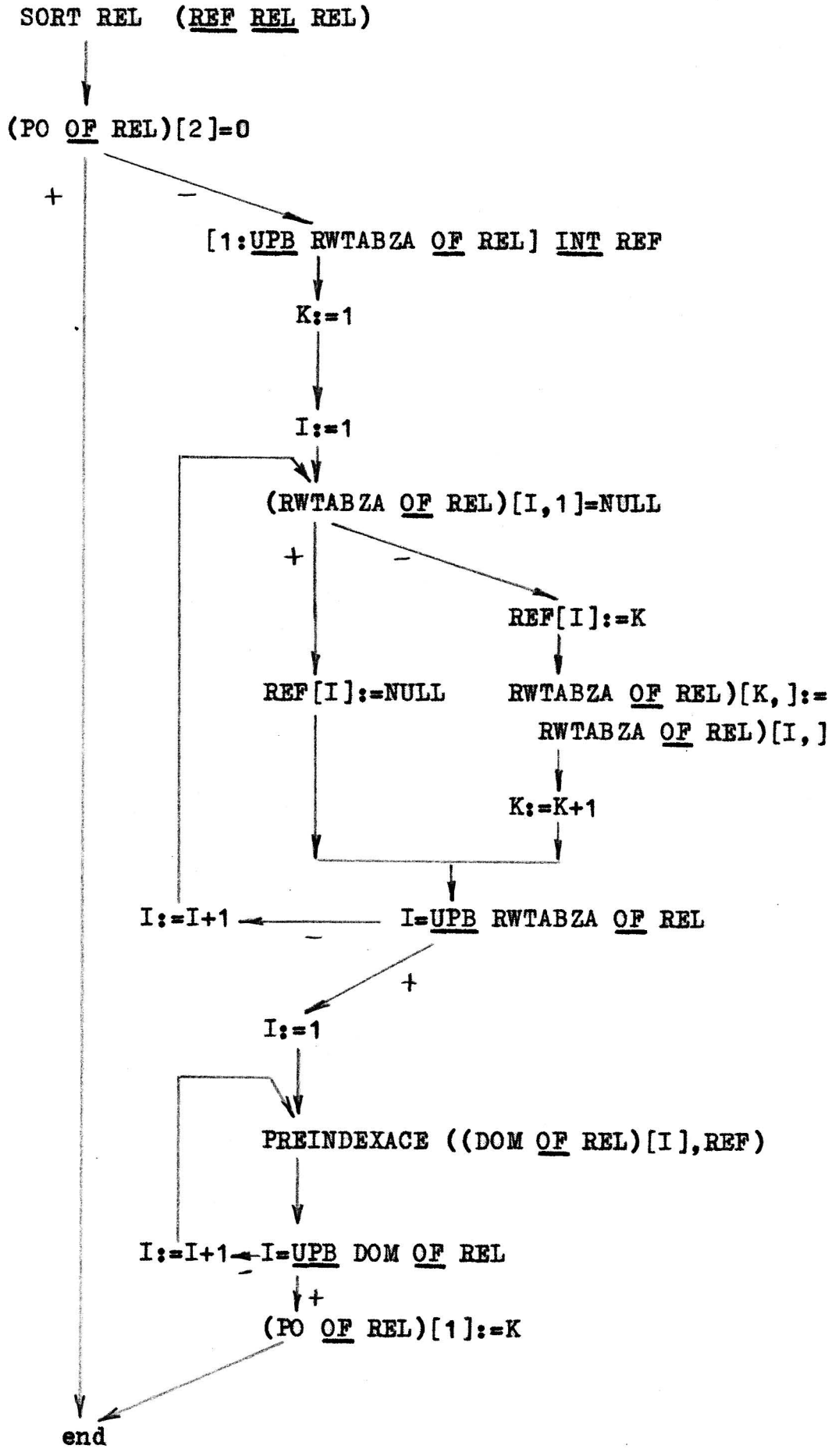

SORT NFD (REF NFD NFD)
BOOL:= FD OF NFD :=: NIL
I:=1
J:=I+1
BOOL
+
-
(VAL OF NFD)[(TAB OF NFD)[I,3]]
> (VAL OF NFD)[(TAB OF NFD)[J,3]]
-
+
GET (FD OF NFD,VAL1,(TAB1 OF NFD)[(TAB OF NFD)[I,3]
GET (FD OF NFD,VAL2,(TAB1 OF NFD)[(TAB OF NFD)[J,3]
VAL1 > VAL2
+
-
Z:=(TAB OF NFD)[I,3]
(TAB OF NFD)[I,3]:=(TAB OF NFD)[J,3]
(TAB OF NFD)[J,3]:=Z
J=(PO OF NFD)[2]-1
J:=J+1
I=(PO OF NFD)[2]-1
I:=I+1
1

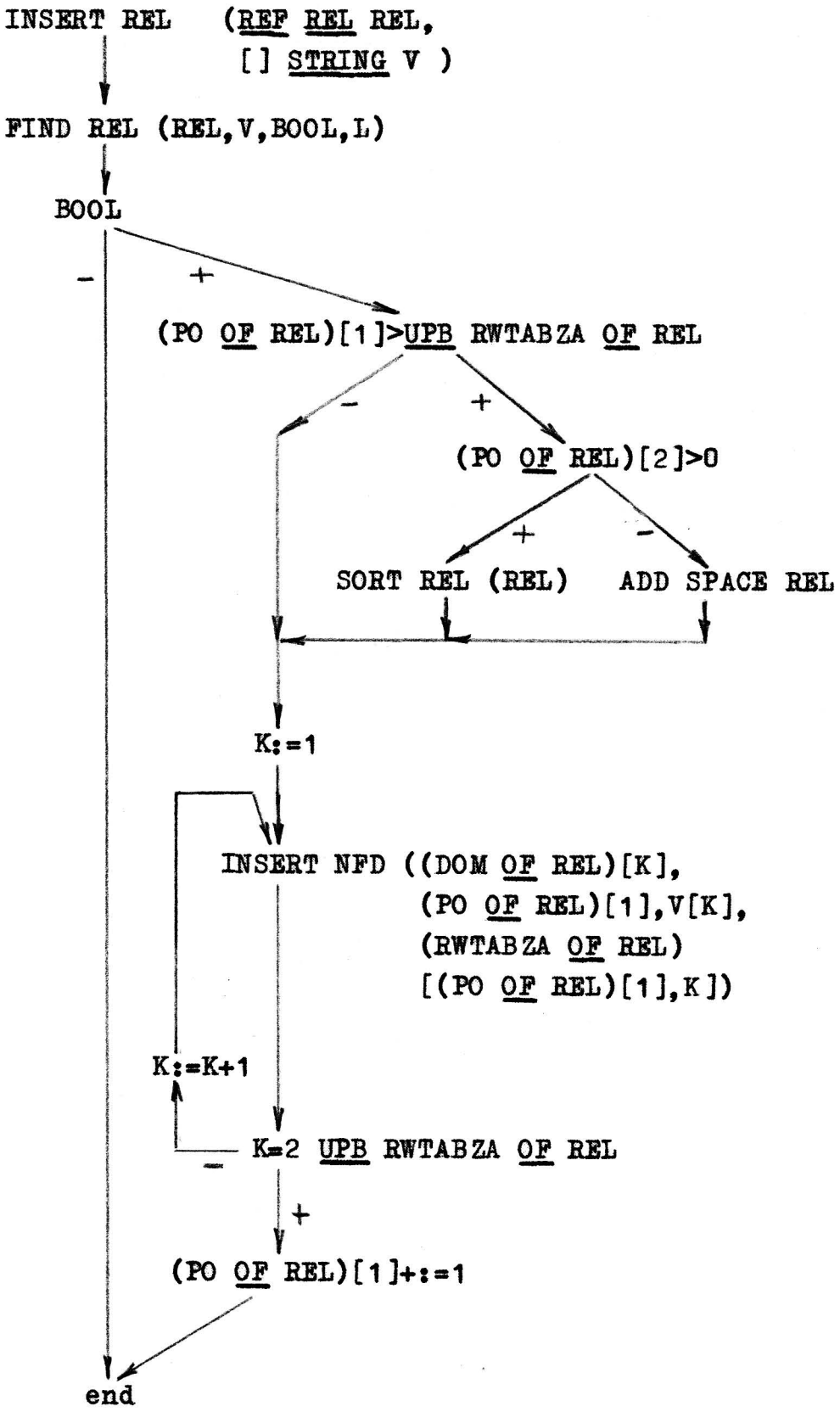

1
(PO OF NFD)[5]=0
I:=1
L:=(TAB OF NFD)[(TAB OF NFD)[I,3],2]
L=0
INT A[1:L]
GET ALL 1 (A,RWTABADR OF NFD,ADR,L1)
RETURN (ADR,RWTABADR OF NFD,CAF OF NFD)
ADR INTO (A[1:L1],RWTABADR OF NFD,ADR,CAF OF NFD)
(TAB OF NFD)[(TAB OF NFD)[I,3],2]:=L1
(TAB OF NFD)[(TAB OF NFD)[I,3],2]:=ADR
I:=I+1
(PO OF NFD)[2]=I
+
(PO OF NFD)[1]:=(PO OF NFD)[2]-:=(PO OF NFD)[3]
(PO OF NFD)[3]:=1
end

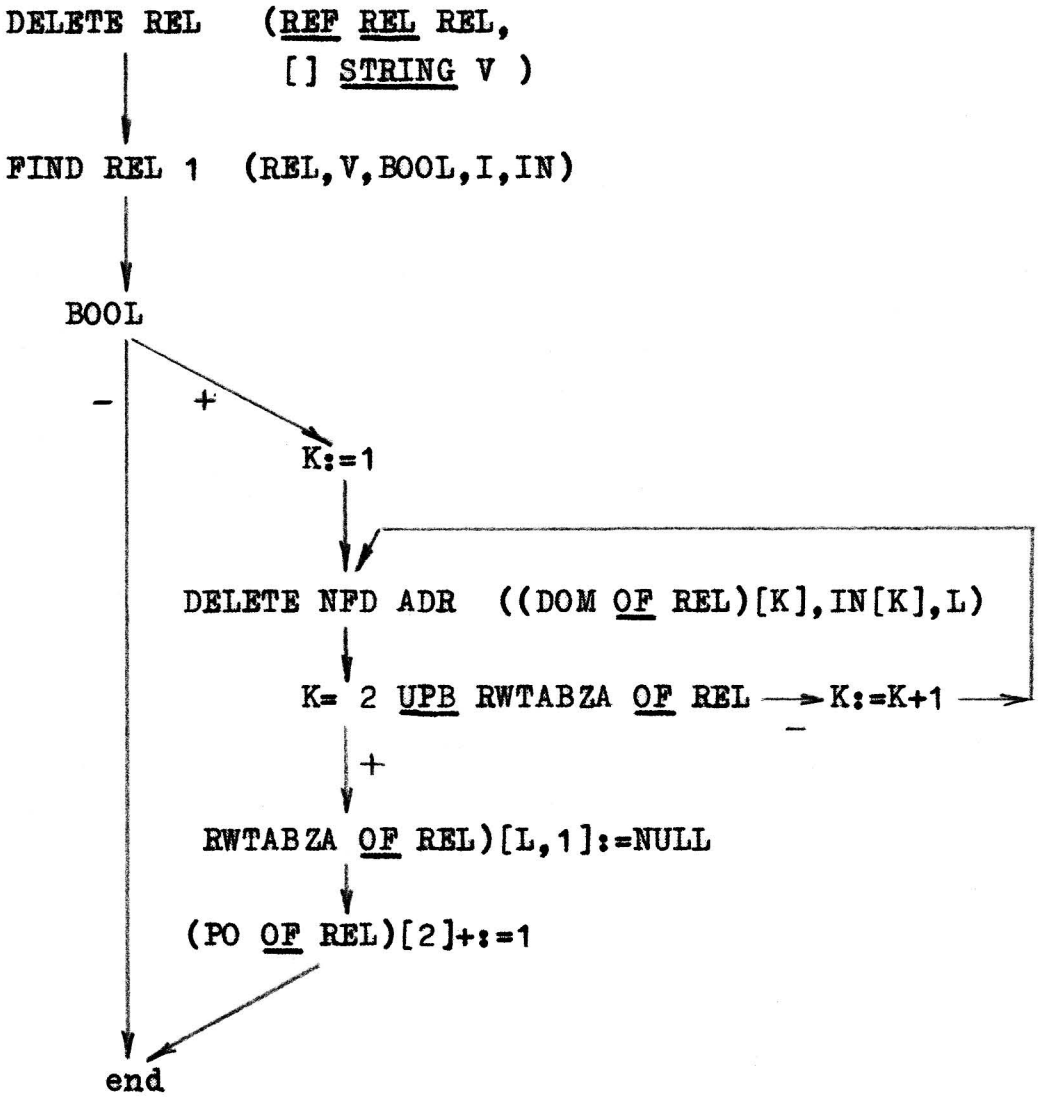
P Ř Í L O H A D

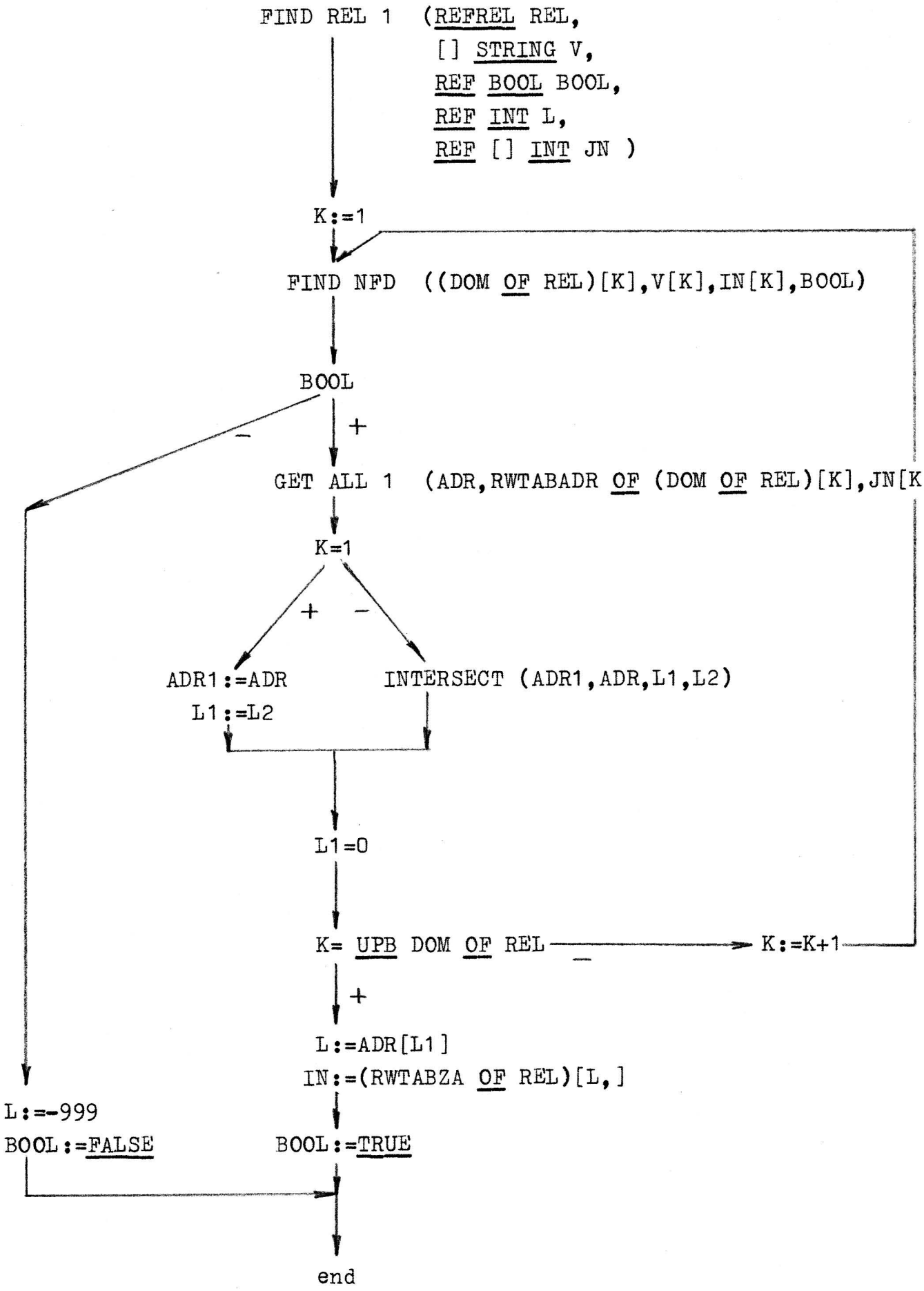
Podrobné vývojové diagramy operací nad rel

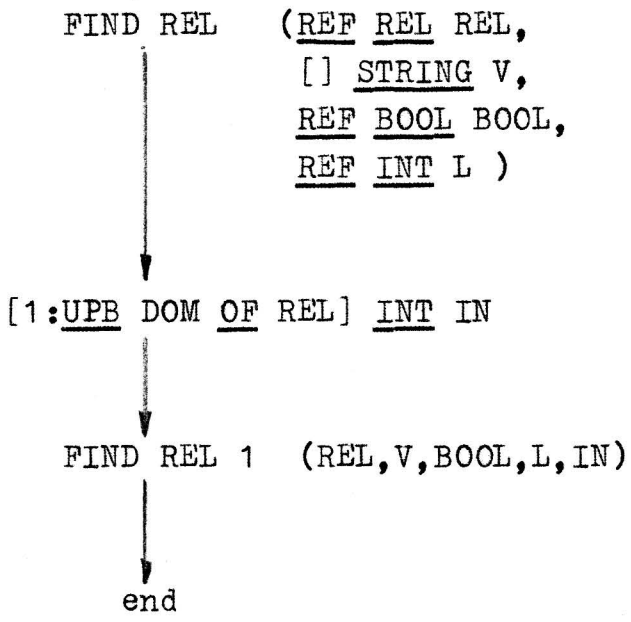

INIT REL (REF REL REL,
[] REF NFD DOM,
INT C1)
DOM OF REL := HEAP [1:UPB DOM] REF NFD := DOM
RWTABZA OF REL:= HEAP [1:C1,1:UPB DOM] INT
I:=1
J:=1
(RWTABZA OF REL)[I,J]:=-1
J=UPB DOM
J:=J+1
+
I=C1
I:=I+1
+
PO OF REL:=(1,0,0)
end

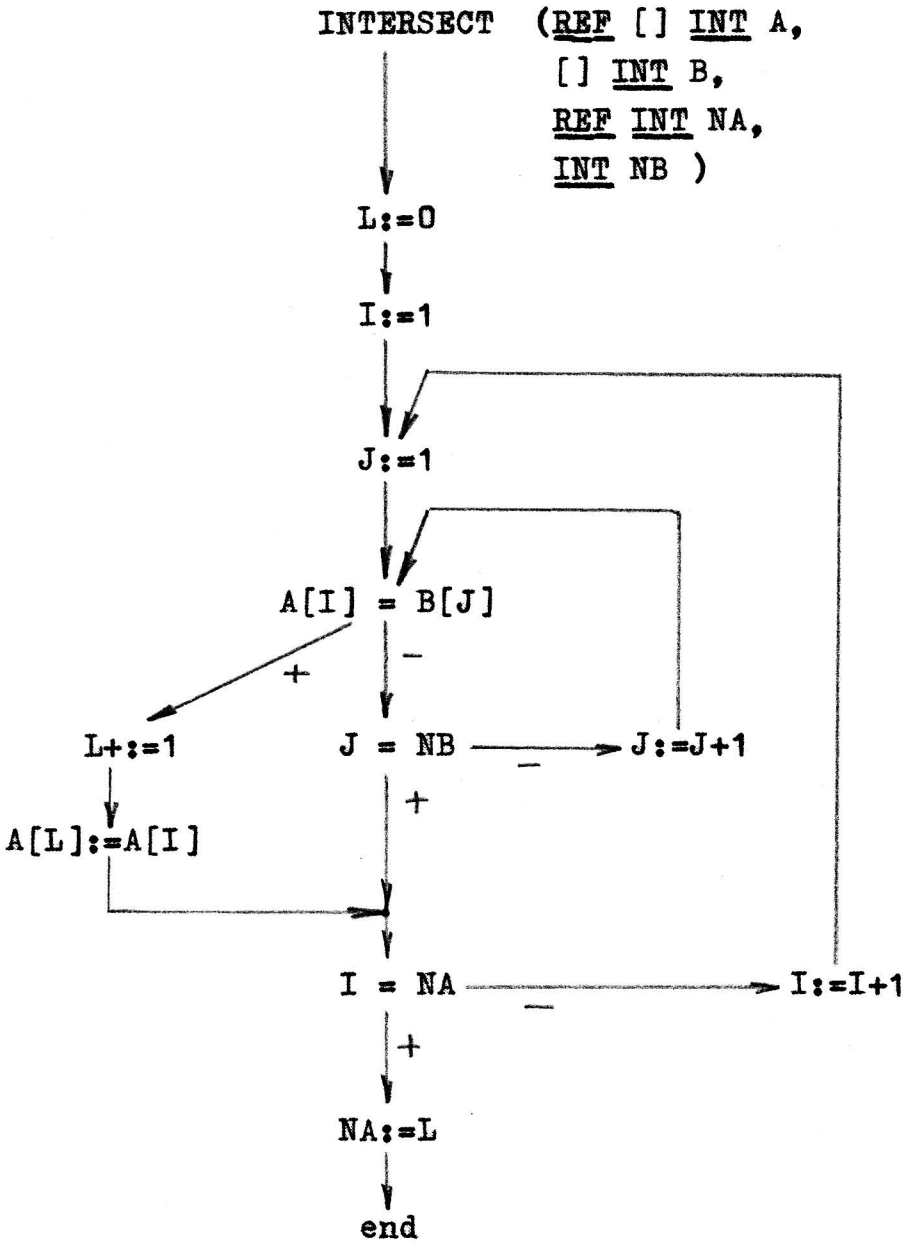

SORT REL (REF REL REL)
(PO OF REL)[2]=0
+
−
[1:UPB RWTABZA OF REL] INT REF
K:=1
I:=1
(RWTABZA OF REL)[I,1]=NULL
+
−
REF[I]:=K
REF[I]:=NULL
RWTABZA OF REL)[K,]:=
RWTABZA OF REL)[I,]
K:=K+1
I:=I+1
I=UPB RWTABZA OF REL
−
+
I:=1
PREINDEXACE ((DOM OF REL)[I],REF)
I:=I+1
I=UPB DOM OF REL
−
+
(PO OF REL)[1]:=K
end


INSERT REL (REF REL REL,
[] STRING V)
FIND REL (REL,V,BOOL,L)
BOOL
-
+
(PO OF REL)[1]>UPB RWTABZA OF REL
-
+
(PO OF REL)[2]>0
+
-
SORT REL (REL)
ADD SPACE REL
K:=1
INSERT NFD ((DOM OF REL)[K],
(PO OF REL)[1],V[K],
(RWTABZA OF REL)
[(PO OF REL)[1],K])
K:=K+1
K=2 UPB RWTABZA OF REL
-
+
(PO OF REL)[1]+:=1
end


DELETE REL (REF REL REL,
[] STRING V)
FIND REL 1 (REL,V,BOOL,I,IN)
BOOL
-
+
K:=1
DELETE NFD ADR ((DOM OF REL)[K],IN[K],L)
K= 2 UPB RWTABZA OF REL
K:=K+1
-
+
RWTABZA OF REL)[L,1]:=NULL
(PO OF REL)[2]+:=1
end


FIND REL 1 (REFREL REL,
[] STRING V,
REF BOOL BOOL,
REF INT L,
REF [] INT JN)
K:=1
FIND NFD ((DOM OF REL)[K],V[K],IN[K],BOOL)
BOOL
+
GET ALL 1 (ADR,RWTABADR OF (DOM OF REL)[K],JN[K],
K=1
+
ADR1:=ADR
L1:=L2
INTERSECT (ADR1,ADR,L1,L2)
L1=0
K= UPB DOM OF REL
K:=K+1
+
L:=ADR[L1]
IN:=(RWTABZA OF REL)[L,]
L:=-999
BOOL:=FALSE
BOOL:=TRUE
end


FIND REL (REF REL REL,
[] STRING V,
REF BOOL BOOL,
REF INT L)
[1:UPB DOM OF REL] INT IN
FIND REL 1 (REL,V,BOOL,L,IN)
end


INTERSECT (REF [] INT A,
[] INT B,
REF INT NA,
INT NB)
L:=0
I:=1
J:=1
A[I] = B[J]
+
−
L+:=1
J = NB
−
J:=J+1
+
A[L]:=A[I]
I = NA
−
I:=I+1
+
NA:=L
end

P Ř Í L O H A E

Kapacitní propočty

Vzhledem k tomu, že použití organizace Master-Detail u souborů není obvyklé, uvedeme nyní odhady spotřeby paměti. Navzájem budou porovnávány následující organizace souboru:

1) Multi-list (vícenásobný seznam)

2) Inverted-list (invertovaný seznam)

3) Master-Detail

4) Relace (struktura navržená v kap.6. bez použití fundamentální oblasti)

V dalším budeme používat následující označení:

n ... počet položek ve větě

l_i ... délka i-té položky

l ... $\frac{1}{n}\sum_{i=1}^{n} l_i$ průměrná délka položky

p_i ...počet různých hodnot i-té položky

p ... $\frac{1}{n}\sum_{i=1}^{n} p_i$ průměrná hodnota počtu různých hodnot položek

m ... počet vět v souboru

t ... počet klíčových položek

C ... požadovaná paměť

Nyní uvedeme odhad paměti vyžadované jednotlivými organizacemi souboru (předpokládáme, že směrník se ukládá na 4 Byty)

Multi-list

a) soubor $m.(n.l+4.t)$

b) adresáře $t.(l+4).p$

Celkem $C_{ML}=m.(n.l+4.t)+t.p.(l+4)$

Inverted-list

a) soubor $C_1 = m.n.l$

b) adresáře $C_2 = t.p.(l+4.m/p)$

Celkem $C_{IL} = m.n.l + 4.t.m + t.l.p$

Master-Detail

a) soubor $C_1 = 8.m.n.$

b) adresáře $C_2 = (l+4).p.n$

Celkem $C_{MD} = 8.m.n + n.p.(l+4)$

Relace

a) soubor $C_1 = 4.m.n$

b) adresáře $C_2 = ((l+12).p + m).n$

Celkem $C_{RE} = 4.m.n + ((l+12).p+m).n$

Vzhledem k tomu, že požadujeme, aby všechny položky vět měly vlastnost klíčových položek, budeme dále uvažovat, že počet klíčů je roven počtu položek, tj. $t = n$.
Pokusme se nyní ukázat, jaké vztahy z hlediska spotřeby paměti jsou mezi jednotlivými organizacemi souboru.

1) Multi-list vůči Inverted-list

$$C = C_{ML} - C_{IL} = m.n.(l+4)+n.p.(l+4)-m.n.l-4.m.n-n.p.l$$

a tedy:

$$C = 4.n.p$$

Potřebná paměť u organizace Multi-list je vždy větší než paměť potřebná u organizace Inverted-list. Rozdíl je závislý na počtu klíčů a na průměrném počtu různých hodnot položky v souboru.

2) Multi-list vůči Master-Detail

$$C = C_{ML} - C_{MD} = m.n.(1+4)+n.p.(1+4)-8.m.n-n.p.(1+4) - m.n.(1-4)$$

a tedy:

$$C = C_{ML}-C_{MD} = m.n.(1-4)$$

Potřeba paměti při organizaci Multi-list je vždy větší než potřeba paměti pro organizaci Master-Detail, je-li průměrná délka položek větší než 4 Byty.

3) Multi-list vůči Relace

$$C = C_{ML}- C_{RE} = m.(n.l+4.n)+n.p.(l+4)-4.m.n-((l+12).p+m).n$$
$$= m.n.l-m.n-8.n.p = n.(m.(l-1)-8.p)$$

Protože $p \leq m$ zvolme nejhorší případ, a to $p = m$. Pak:

$$n.(m.(l-1)-8.p) \leq m.n.(l-9)$$

a tedy :

$$C = C_{ML} - C_{RE} \leq m.n.(l-9)$$

Paměť potřebná u organizace Multi-list je vždy větší než u organizace Relace v případě, že průměrná délka položky je větší než 9 Bytů.

4) Inverted-list vůči Master-Detail

$$C = C_{IL} - C_{MD} = m.n.l + n.l.p + 4.n.m -8.m.n-n.p.l-4.n.p$$
$$= m.n.l - 4.m.n - 4.n.p$$

Platí, že $p \leq m$, a proto zvolme opět nejhorší případ, tj. $p = m$. Pak:

$$C = C_{IL} - C_{MD} \leq m.n.(1-8)$$

Paměť potřebná u organizace Inverted-list je vždy větší než u organizace Master-Detail v případě, že průměrná délka položek je větší než 8 Bytů.

5) Inverted-list vůči Relace

$$C = C_{IL} - C_{RE} = m.n.l + n.p.l + 4.m.n - 4.m.n - ((1+12).p+m).n = m.n.l - m.n - 12n.p$$

Platí, že $p \leq m$, a proto zvolme opět nejhorší případ, tj. $p = m$.

Pak platí:

$$C = C_{IL} - C_{RE} = m.n.(1-13)$$

Paměť potřebná u organizace Inverted-list je vždy větší než u organizace Relace, je-li průměrná délka položky větší než 13 Bytů.

6) Master-detail vůči Relace

$$C = C_{MD} - C_{RE} = 8.m.n. + n.p.(1+4) - 4.m.n - ((1+12).p+m).n = = 3\ m.n - 8n.p = n.(3.m - 8p)$$

Je-li $3m - 8p < 0$, pak je výhodnější použít Relace.

Pro přehlednost uveďme ještě tabulku rozdílů potřebné paměti u jednotlivých implementací s podmínkami platnosti. Tabulka vyjadřuje vztah $C_A - C_B \; ? \; 0$, za předpokladu, že počet klíčových položek $t = n$.

C_A \ C_B	ML	IL	MD	RE
ML	-	$>$	$l \overset{>}{>} 4$	$l > 9, \overset{>}{} p \leq m$
IL	$<$	-	$l > 8, \overset{>}{} p \leq m$	$l > 13, \overset{>}{} p \leq m$
MD	$l \overset{<}{>} 4$	$l > 8, \overset{<}{} p \leq m$	-	$p \overset{>}{>} \frac{3}{8} m$
RE	$l > 9, \overset{<}{} p \leq m$	$l > 13, \overset{<}{} p \leq m$	$p \overset{<}{>} \frac{3}{8} m$	-

tab.E.1

Ve výše uvedených případech jsme se zabývali případem, kdy každá položka je klíčová a má navíc vlastnost primárního klíče, tj. p = m. V naprosté většině případů však bude $p < m$ a také lze předpokládat, že t n. Pak:

$$C_{ML} - C_{RE} = m.n.l + 4 + m + tpl + 4tp - 4mn - lpn - 12pn - mn =$$

$$= m.n\left[1+4\frac{t}{n}+ 1\frac{t}{n}\frac{p}{m} + 4\frac{t}{n}\frac{p}{m} - 4 - 1\frac{p}{m} - 12\frac{p}{m} - 1\right] = 0$$

pro $m \neq 0 \wedge n \neq 0$ dostáváme:

$$\frac{p}{m}\left[1\frac{t}{n} + 4\frac{t}{n} - 1 - 12\right] + \left[1 + 4\frac{t}{n} - 5\right] = 0$$

Pak vztah:

$$\frac{p}{m} = \frac{1 + 4\frac{t}{n} - 5}{1 + 12 - (1+4)\frac{t}{n}}$$

Závislost je vyjádřena tabulkou tab.E.2 a je na obr.E.1.

Analogickým postupem pro :

$$C_{IL} - C_{RE} = 0$$

dostáváme :

$$\frac{p}{m} = \frac{\sqrt{1 + 4\frac{t}{n}} - 5}{1 + 12 - 1\frac{t}{n}}$$

Závislost je vyjádřena tabulkou tab.E.3. a je na obr.E.2.

KAPACITNÍ PROPOCTY

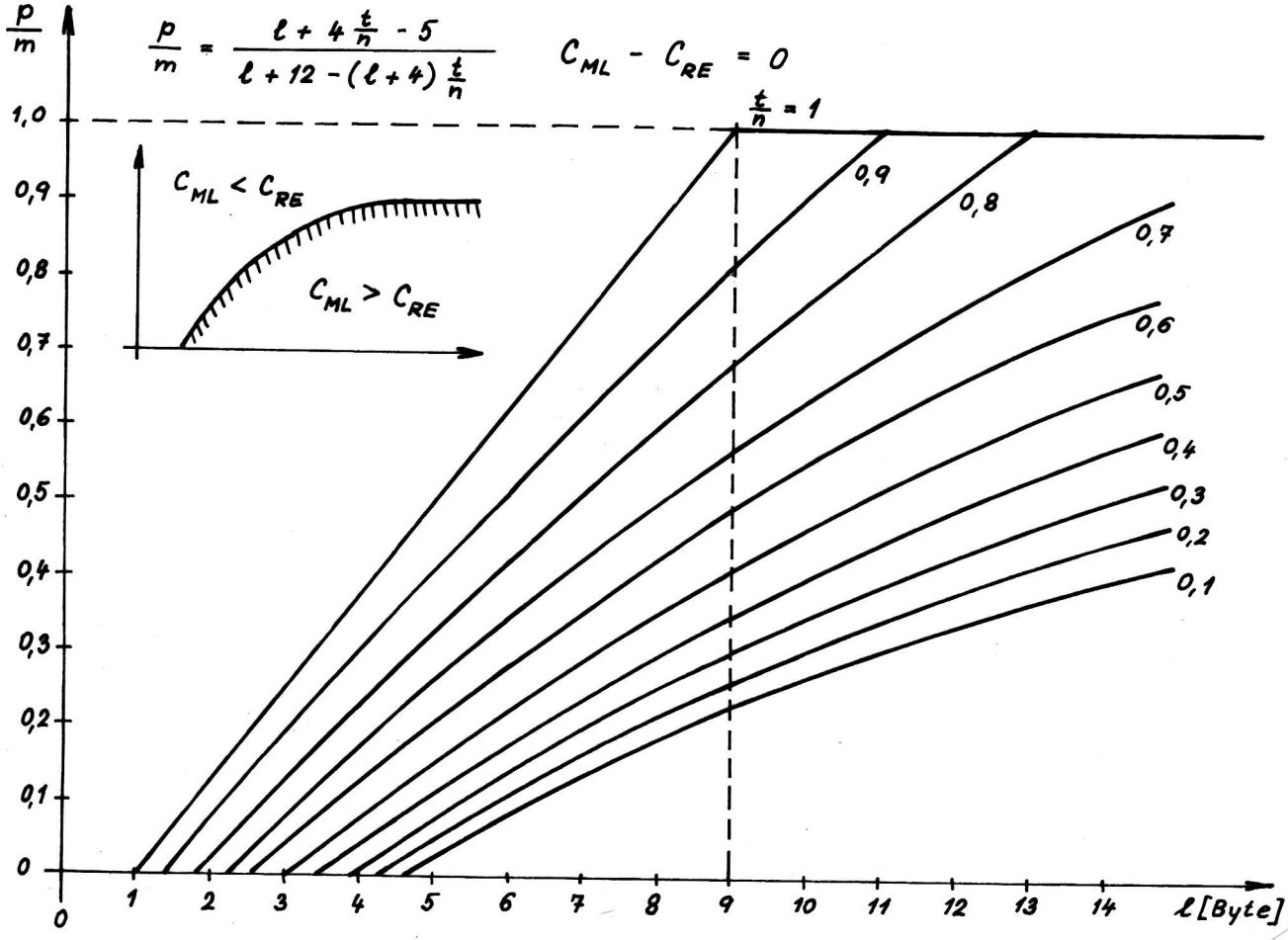
l \ t/n	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1	-0.29	-0.27	-0.24	-0.22	-0.19	-0.16	-0.13	-0.09	-0.05	0.00
2	-0.19	-0.17	-0.15	-0.12	-0.09	-0.06	-0.02	0.02	0.07	0.13
3	-0.11	-0.09	-0.06	-0.03	0.00	0.04	0.08	0.13	0.18	0.25
4	-0.04	-0.01	0.01	0.05	0.08	0.13	0.17	0.23	0.30	0.38
5	0.02	0.05	0.08	0.12	0.16	0.21	0.26	0.33	0.40	0.50
6	0.08	0.11	0.15	0.19	0.23	0.28	0.35	0.42	0.51	0.63
7	0.13	0.17	0.20	0.25	0.30	0.35	0.42	0.51	0.62	0.75
8	0.18	0.22	0.26	0.30	0.36	0.42	0.50	0.60	0.72	0.88
9	0.22	0.26	0.30	0.35	0.41	0.48	0.57	0.68	0.82	1.00
10	0.26	0.30	0.35	0.40	0.47	0.54	0.64	0.76	0.91	1.13
11	0.30	0.34	0.39	0.45	0.52	0.60	0.70	0.84	1.01	1.25
12	0.33	0.38	0.43	0.49	0.56	0.65	0.77	0.91	1.10	1.38
13	0.36	0.41	0.46	0.53	0.61	0.70	0.82	0.98	1.20	1.50

tab.E.2.

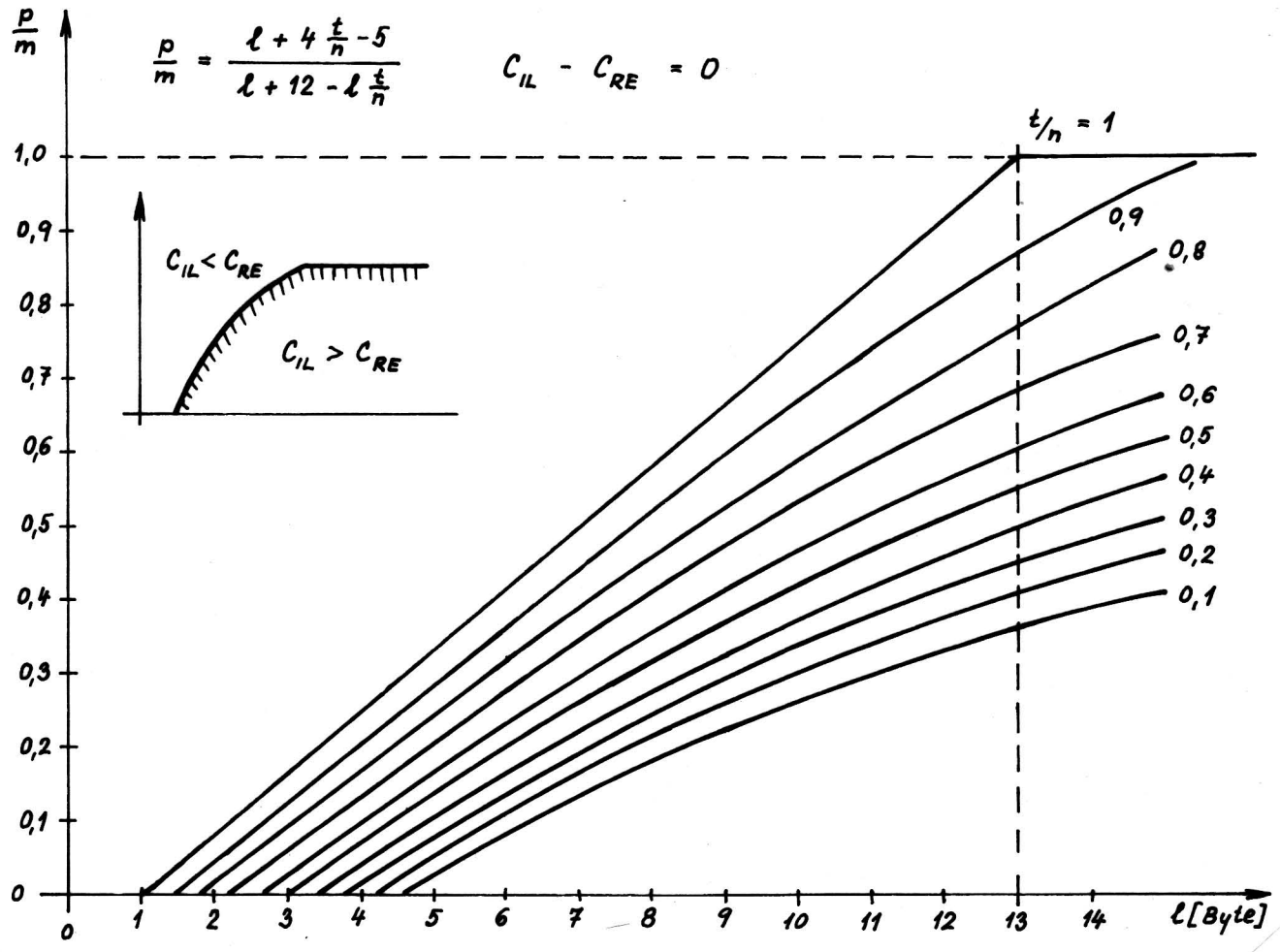
KAPACITNÍ PROPOCTY

l \ t/n	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1	-0.28	-0.25	-0.22	-0.19	-0.16	-0.13	-0.10	-0.07	-0.03	0.00
2	-0.19	-0.16	-0.13	-0.11	-0.08	-0.05	-0.02	0.02	0.05	0.08
3	-0.11	-0.08	-0.06	-0.03	0.00	0.03	0.06	0.10	0.13	0.17
4	-0.04	-0.01	0.01	0.04	0.07	0.10	0.14	0.17	0.21	0.25
5	0.02	0.05	0.08	0.11	0.14	0.17	0.21	0.25	0.29	0.33
6	0.08	0.11	0.14	0.17	0.20	0.24	0.28	0.32	0.37	0.42
7	0.13	0.16	0.19	0.22	0.26	0.30	0.34	0.39	0.44	0.50
8	0.18	0.21	0.24	0.27	0.31	0.36	0.40	0.46	0.52	0.58
9	0.22	0.25	0.28	0.32	0.36	0.41	0.46	0.52	0.59	0.67
10	0.26	0.29	0.33	0.37	0.41	0.46	0.52	0.59	0.66	0.75
11	0.29	0.33	0.37	0.41	0.46	0.51	0.58	0.65	0.73	0.83
12	0.32	0.36	0.40	0.45	0.50	0.56	0.63	0.71	0.80	0.92
13	0.35	0.39	0.44	0.48	0.54	0.60	0.68	0.77	0.87	1.00

tab.E.3.


p/m
p/m = (l + 4 t/n - 5) / (l + 12 - (l + 4) t/n)
C_ML - C_RE = 0
t/n = 1
C_ML < C_RE
C_ML > C_RE
1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
1 2 3 4 5 6 7 8 9 10 11 12 13 14
l [Byte]

obr. E.1.


p/m
p/m = (l + 4 t/n - 5) / (l + 12 - l t/n)
C_IL - C_RE = 0
t/n = 1
1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
C_IL < C_RE
C_IL > C_RE
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
l[Byte]

obr. E.2.

P Ř Í L O H A F

Programové řešení operací nad fd a nfd

```
LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

  1.   'BEGIN
  2.    'INT HIGH:=MAXINT,NULL:=MAXINT;                                                          B
  3.  # DEFINICE MODU FUNDAMENTALNI OBLASTI #                                                    B
  4.   'MODE'FD ='STRUCT (                                                                       B
  5.                       'REF []'INT REF,                                                     B(
  6.                       'REF [,]'CHAR VAL,                                                   B(
  7.                       [1..4]'INT PO);                                                      B(
  8.                                                                                            B
  9.   'PROC SORT=     ('REF'FD FUDO)'VOID :                                                    B
 10.  #**********#                                                                              B
 11.   'BEGIN                                                                                   B
 12.     'INT   POCET,K,J,I1,J1,I,L;                                                            BB
 13.     'STRING V1,V2;                                                                         BB
 14.     POCET:=(PO'OF FUDO)[3]-1;                                                              BB
 15.     # TRIDENI TABULKY FUDO PODLE REF ,HODNOTY VAL ZUSTAVAJI #                              BB
 16.           PRINT(("    S O R T  ",NEWLINE));                                              BB
 17.     'FOR I   'TO (PO'OF FUDO)[2]-1'DO                                                      BB
 18.   'FOR J'FROM  I+1'TO (PO'OF FUDO)[2]-1'DO                                                 BBD
 19.           I1:=(REF'OF FUDO)[I];                                                          BBDD
 20.           J1:=(REF'OF FUDO)[J];                                                          BBDD
 21.                          V1:=(VAL'OF FUDO)[I1,];                                         BBDD
 22.                           V2:=(VAL 'OF FUDO)[J1,];                                       BBDD
 23.           'IF V1'GT V2 'THEN                                                             BBDD
 24.              K:=(REF'OF FUDO)[I];                                                        BBDDI
 25.              (REF'OF FUDO)[I]:=(REF'OF FUDO)[J];                                         BBDDI
 26.              (REF'OF FUDO)[J]:=K                                                         BBDDI
 27.           'FI                                                                            BBDDI
 28.           # ELSE=PRVKY JSOU VE SPRAVNEM PORADI #                                        BBDD
 29.        'OD                                                                             BBDD
 30.     'OD;                                                                               BBD
 31.     (PO'OF FUDO)[2]:=(PO'OF FUDO)[2]-POCET;                                            BB
 32.     (PO'OF FUDO)[1]:=(PO'OF FUDO)[2];                                                  BB
 33.       (PO'OF FUDO)[3]:=1                                                               BB
 34.     # TABULKY VINDEX A REF SETRIDENY #                                                  BB
 35.   # END OF PROC SORT #                                                                 BB
 36.   'END;                                                                                BB
 37.                                                                                        B
 38.   'PROC FIND =    ('REF'FD FD  , []'CHAR VALUE,                                        B
 39.  #**********#                                                                          B(
 40.                    'REF'BOOL BOOL,'REF'INT IN)'VOID:                                   B(
 41.   # PROCEDURA ZAJISTI VYHLEDANI HODNOTY VE FUNDAMENTALNI  #                            B
 42.   # ZONE                                                  #                            B
 43.   # VSTUPNI PARAMETRY;                                    #                            B
 44.   #     FUDO  PROMENNA MODU FD -FUNDAMENTALNI OBLAST      #                            B
 45.   #     VALUE HODNOTA,KTEROU CHCEME NAJIT                 #                            B
 46.   # VYSTUPNI PARAMETRY:                                   #                            B
 47.   #     BOOL  JE HODNOTY TRUE, NALEZLA-LI SE POZADOVANA   #                            B
 48.   #              HODNOTA,JINAK JE FALSE                   #                            B
 49.   #     IN    V PRIPADE,ZE BOLL=TRUE,PAK IN UDAVA CISLO    #                           B
 50.   #        NA HODNOTU                                        #                         B

LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP
```

```
LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

  51.   'BEGIN                                                                                    B
  52.    'INT D:=0,I,J,K;                                                                       BB
  53.   BOOL:='FALSE;                                                                           BB
  54.  IN:=NULL;                                                                                BB
  55.  'IF (PO'OF FD)[1]'NE 1'THEN                                                               BB
  56.  'IF (VAL'OF FD)[(REF'OF FD)[(PO'OF FD)[1]-1],] 'GE VALUE 'THEN                            BBI
  57.        I:=1;J:=0;K:=1                                                                     BBII
  58.      'ELSE D:=(PO'OF FD)[1]'FI                                                            BBII
  59.      'ELSE D:=(PO'OF FD)[1]'FI;                                                           BBI
  60.  'WHILE D<(PO'OF FD)[1]-1'DO                                                             BB
  61.  'IF (VAL'OF FD)[(REF'OF FD)[D+I+J          ],]'LT  VALUE 'THEN                           BBD
  62.            K:=I;                                                                        BBDI
  63.            I:=I+J;                                                                      BBDI
  64.            J:=K;                                                                        BBDI
  65.            'IF D+J+I > (PO'OF FD)[1]-1 'THEN                                            BBDI
  66.            D:=D+I;                                                                      BBDII
  67.            I:=K:=1;                                                                     BBDII
  68.            J:=0                                                                         BBDII
  69.            'FI                                                                          BBDII
  70.            'ELSE                                                                        BBDI
  71.  'IF (VAL'OF FD)[(REF'OF FD)[D+I+J          ],] =   VALUE 'THEN                           BBDI
  72.                     BOOL:='TRUE;                                                        BBDII
  73.                     IN:=(REF'OF FD)[D+I+J];                                             BBDII
  74.  'GOTO FINDR                                                                           BBDII
  75.            'ELSE                                                                        BBDII
  76.            D:=D+I;                                                                      BBDII
  77.            I:=K:=1;                                                                     BBDII
  78.            J:=0                                                                         BBDII
  79.            'FI                                                                          BBDII
  80.            'FI                                                                          BBDI
  81.            'OD;                                                                         BBD
  82.  STX:                                                                                    BB
  83.   # HODNOTA NENI V SETRIDENE ZONE #                                                      BB
  84.     'IF (PO'OF FD  )[2]'NE (PO'OF FD  )[1]'THEN                                         BB
  85.      'FOR J'FROM (PO'OF FD  )[1]'TO (PO'OF FD  )[2]-1'DO                                BBI
  86.          'IF (VAL'OF FD  )[(REF'OF FD  )[J],]=VALUE                                     BBID
  87.            'THEN                                                                        BBIDI
  88.               BOOL:='TRUE;                                                              BBIDI
  89.            IN:=(REF'OF FD)[J];                                                          BBIDI
  90.               'GOTO FINDR                                                               BBIDI
  91.         'FI                                                                            BBIDI
  92.        'OD;                                                                            BBID
  93.        BOOL:='FALSE                                                                    BBI
  94.       # HODNOTA NENI ANI V NESETRIDENE ZONE #                                          BBI
  95.     'FI;                                                                              BBI
  96.   FINDR:'SKIP                                                                         BB
  97.   # END OF PROC FIND #                                                                BB
  98.   'END;                                                                               BB
  99.                                                                                       B
 100.   'PROC INSERT=    ('REF'FD FUDO, []'CHAR VALUE,                                      B

LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP
```

```
LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

101.                    'REF'INT IN,'REF 'BOOL BOOL)'VOID:                                      B(
102.   'BEGIN                                                                                   B
103.       FIND(FUDO,VALUE,BOOL,IN);                                                           BB
104.       'IF 'NOT BOOL 'THEN                                                                 BB
105.       'IF (PO 'OF FUDO)[2] 'GT 'UPB REF 'OF FUDO 'THEN                                    BBI
106.       'IF (PO 'OF FUDO)[3]=1 'THEN                                                        BBII
107.            PRINT(("POLOZKA  ",VALUE,"  NEMA  MISTO ",NEWLINE));                          BBIII
108.            'GOTO HTX                                                                      BBIII
109.                           'ELSE SORT(FUDO)                                                BBIII
110.       'FI                                                                                 BBIII
111.       'FI;                                                                                BBII
112.            (VAL'OF FUDO)[(REF'OF FUDO)[(PO'OF FUDO)[2]],]:=VALUE;                         BBI
113.          (PO'OF FUDO)[2]:=(PO'OF FUDO)[2]+1                                               BBI
114.          # POLOZKA PREDTÍM NEBYLA VE SLOVNÍKU #                                           BBI
115.       'ELSE                                                                               BBI
116.        PRINT((NEWLINE,"POLOZKA",VALUE,"BYLA ZALOZENA DRIVE DO FD",NEWLINE                BBI
117.                                                                          ))               BBI((
118.       'FI                                                                                 BBI
119.  ; HTX:'SKIP                                                                              BB
120.            'END;                                                                         BB
121.   'PROC GET  =     ('REF'FD FUDO,'REF[]'CHAR VALUE,                                       B
122.                     'INT I )'VOID :                                                       B(
123.   'BEGIN                                                                                  B
124.       'IF I 'GT 'UPB REF 'OF FUDO 'THEN                                                   BB
125.        PRINT((NEWLINE,"PRILIS VELKA HODNOTA I",I,NEWLINE))                                BBI
126.       'ELSE                                                                               BBI
127.                              VALUE:=(VAL'OF FUDO)              [I,]                      BBI
128.       'FI                                                                                 BBI
129.   'END;                                                                                   BB
130.   'PROC DELETE=      ('REF'FD FUDO,                                                       B
131.                       'INT IN)'VOID :                                                     B(
132.   'BEGIN'INT I;                                                                           B
133.  'INT L=2'UPB VAL'OF FUDO;                                                               BB
134.  [1..L]'CHAR SNULL;                                                                      BB
135.  'FOR I 'TO L'DO                                                                         BB
136.            SNULL[I]:="←"                                                                 BBD
137.  'OD;                                                                                    BBD
138.       'IF IN 'GT 'UPB REF 'OF FUDO 'THEN                                                  BB
139.        PRINT((NEWLINE,"PRILIS VELKA HODNOTA I ",IN,NEWLINE))                              BBI
140.       'ELSE                                                                               BBI
141.           (VAL 'OF FUDO)[                IN,] :=SNULL;                                  BBI
142.                     (PO 'OF FUDO)[3]:=(PO'OF FUDO)[3]+1                                  BBI
143.                  'FI                                                                     BBI
144.   'END;                                                                                   BB
145.   # PROCEDURA PRO TISK FUNDAMENTALNI OBLASTI #                                           B
146.   'PROC TISK FD=     ('FD FD)'VOID:                                                       B
147.   (    'INT K:=7                                                                        B
148.     ; PRINT((NEWLINE,"TISK HODNOT FUNDAMENTALNI OBLASTI",NEWLINE))                      B(
149.     ; PRINT((NEWLINE,"POLE PO",NEWLINE))                                                B(
150.              ;   PRINT       ((PO'OF FD                                                B(

LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP
```

```
LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

151.                    ,NEWLINE,NEWLINE))                                                      B(((
152.     ; PRINT(("          ",                                                                 B(
153.              "C.RADKU    ",        " REF       VAL",NEWLINE))                             B(((
154.     ; 'FOR I'TO'UPB REF'OF FD'DO                                                           B(
155.    PRINT( (I,                                  (REF'OF FD)[I]                              B(D
156.  ,"     " ,(VAL'OF FD)[I,],"          ",                                                  B(D((
157.                        (VAL'OF  FD)[(REF'OF FD)[I],],                                      B(D((
158.                        NEWLINE))                                                           B(D((
159.       'OD                                                                                  B(D
160.    ) ;                                                                                     B(
161.    'PROC SRAND=     ('INT POCET)[]'CHAR:                                                   B
162.    # PROCEDURA PRO GENERACI NAHODNYCH RETEZCU DELKY 7 ZNAKU #                              B
163.    (   'STRING S:=""                                                                       B
164.        ; 'REAL RA                                                                          B(
165.      ; 'FOR I'TO  POCET 'DO                                                                B(
166.       RA:=RANDOM;                                                                          B(D
167.        S:=S+ 'CASE 1+'ENTIER   ( RA    *12)                                              B(D
168.                  'IN "A","B","C","D","E","F",                                            B(DC
169.                      "G","H","I","J","K","L"                                             B(DC
170.                  'OUT "Z"                                                                B(DC
171.              'ESAC                                                                       B(DC
172.        'OD                                                                               B(D
173.      ; S                                                                                 B(
174.    ) ;                                                                                   B(
175.    'PROC ZKUS FD=    ('REF'FD FUDO)'VOID:                                                B
176.    (  'STRING S                                                                          B
177.      ; 'INT N ;'BOOL BOOL;                                                               B(
178.  'INT L=2'UPB VAL'OF FUDO;                                                               B(
179.  [1..L]'CHAR SNULL;                                                                     B(
180.  'FOR I 'TO L'DO                                                                        B(
181.            SNULL[I]:="↑"                                                                 B(D
182.  'OD;                                                                                    B(D
183.       'FOR I 'TO 7 'DO                                                                   B(
184.           'FOR J'TO 10'DO                                                                B(D
185.                S:=SRAND( L)                                                              B(DD
186.             ;  INSERT(FUDO,S,N,BOOL)                                                     B(DD
187.           'OD                                                                            B(DD
188.         ; TISK FD(FUDO)                                                                  B(D
189.         ; SORT (FUDO)                                                                    B(D
190.         ; TISK FD(FUDO)                                                                  B(D
191.       'OD                                                                                B(D
192.      ;  DELETE(FUDO,10)                                                                  B(
193.     ; INSERT(FUDO,SNULL,N,BOOL)                                                          B(
194.     ; INSERT(FUDO,SNULL,N,BOOL)                                                          B(
195.     ;  GET (FUDO,S,5)                                                                    B(
196.     ;  PRINT((NEWLINE,"NALEZENA POLOZKA PRIKAZEM GET ",S,NEWLINE))                       B(
197.     ; FIND(FUDO,  SNULL      ,BOOL,N)                                                    B(
198.     ;  'IF 'NOT                                                                         B(
199.           BOOL'THEN PRINT(("POLOZKA NENI V FD",NEWLINE))                                B(I
200.                'ELSE PRINT(("POLOZKA JE V ZONE,INDEX=",N,NEWLINE))                      B(I

LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP
```

```
LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

201.        'FI                                                                                  B(I
202.      ;  SORT(FUDO)                                                                         B(
203.      ; FIND(FUDO,  SNULL     ,BOOL,N)                                                      B(
204.      ; 'IF 'NOT BOOL 'THEN PRINT(("POLOZKA NENI VE FD",NEWLINE))                           B(
205.                      'ELSE PRINT(("POLOZKA JE V ZONE,INDEX= ",N,NEWLINE))                 B(I
206.        'FI                                                                                  B(I
207.      ;  TISK FD(FUDO)                                                                       B(
208.      ;  DELETE(FUDO,N)                                                                     B(
209.      ;  SORT(FUDO)                                                                         B(
210.      ;  TISK FD(FUDO)                                                                      B(
211.    ;                                                                                       B(
212.    'FOR I 'TO 'UPB REF'OF FUDO 'DO                                                         B(
213.               GET(FUDO,S,I);                                                               B(D
214.               PRINT((S,"   "));                                                           B(D
215.               FIND(FUDO,S,BOOL,N)                                                          B(D
216.      ;  'IF 'NOT                                                                          B(D
217.              BOOL'THEN PRINT(("POLOZKA NENI V FD",NEWLINE))                               B(DI
218.                   'ELSE PRINT(("POLOZKA JE V ZONE,INDEX=",N,NEWLINE))                     B(DI
219.      ;  DELETE(FUDO,N)                                                                    B(DI
220.         'FI                                                                               B(DI
221.      ; INSERT(FUDO,S     ,N,BOOL)                                                         B(D
222.    'OD                                                                                     B(D
223.     ; PRINT(("VYPOCET",NEWLINE))                                                          B(
224.     ) ;                                                                                   B(
225.    'PROC WRITE FD=([]'STRUCT('REF'FD FD,[]'CHAR NAME)A)'VOID:                             B
226.    'BEGIN 'FILE FDIN,FDOUT;                                                               B
227.           [1..6]'CHAR NA;                                                                 BB
228.           OPEN(FDOUT,"L9",MT0 CHANN F);                                                   BB
229.           'FOR I'TO 'UPB A'DO                                                             BB
230.               NA:=NAME'OF A[I];                                                           BBD
231.               PUT BIN(FDOUT,                                                              BBD
232.                  (NA,'UPB REF'OF(FD'OF A[I]),                                             BBD(
233.                     2'UPB VAL'OF(FD'OF A[I]),                                             BBD((
234.                           PO'OF(FD'OF A[I]),                                              BBD((
235.                          REF'OF(FD'OF A[I]),                                              BBD((
236.                          VAL'OF( FD'OF A[I])))                                            BBD((
237.           'OD;                                                                            BBD
238.           OPEN(FDIN,"L8",MT1 CHANN F);                                                    BB
239.           ON LOGICAL FILE END (FDIN,('REF'FILE H) 'BOOL:                                  BB
240.               (CLOSE(FDIN);CLOSE(FDOUT);'GOTO EXIT;'TRUE ));                              BB(
241.           'DO                                                                             BB
242.             'FD C;                                                                        BBD
243.             'INT C1,C2;                                                                   BBD
244.              GET BIN(FDIN,                                                                BBD
245.                   (NA,C1,C2,PO'OF C));                                                    BBD(
246.              VAL'OF C:='LOC[1..C1,1..C2]'CHAR;                                            BBD
247.              REF'OF C:='LOC[1..C1]'INT;                                                   BBD
248.              GET BIN(FDIN,(REF 'OF C,VAL'OF C));                                          BBD
249.              PUT BIN(FDOUT,(NA,C1,C2,PO'OF C,REF'OF C,VAL'OF C))                          BBD
250.           'OD;                                                                            BBD

LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP
```

```
LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

  251.   EXIT: 'SKIP                                                                          BB
  252.   'END;                                                                               BB
  253.   'PROC READ FD=('REF'FD FD,[]'CHAR NAME)'VOID:                                       B
  254.   ( 'FILE FDIN;                                                                       B
  255.     [1..6]'CHAR NA;'FD A; 'INT C1,C2;                                                 B(
  256.     OPEN(FDIN,"L9",MT1CHANN F );                                                      B(
  257.     ON LOGICAL FILE END(FDIN,('REF'FILE H)'BOOL:(CLOSE(FDIN);                         B(
  258.               PRINT(("SOUBOR L9 VYCERPAN, FD  ",NAME, " NENI " ,NEWLINE));           B(((
  259.               'GOTO EXIT;'TRUE));                                                     B(((
  260.   L:  GET BIN(FDIN,(NA,C1,C2,PO'OF   A));                                           B(
  261.       'IF C1>'UPB REF'OF FD 'OR C2>2'UPB VAL'OF FD'THEN                             B(
  262.            VAL'OF A:='LOC[1..C1,1..C2]'CHAR;                                        B(I
  263.            REF'OF A:='LOC[1..C1]'INT;                                               B(I
  264.            GET BIN(FDIN,(REF'OF A,VAL'OF A))                                        B(I
  265.          'ELSE                                                                      B(I
  266.            GET BIN(FDIN,(REF'OF  FD,VAL'OF FD))                                     B(I
  267.       'FI;                                                                          B(I
  268.       'IF NA'NE  NAME 'THEN  'GOTO L                                                B(
  269.       'FI;                                                                          B(I
  270.       CLOSE(FDIN);                                                                  B(
  271.   EXIT : 'SKIP                                                                      B(
  272.   );                                                                                B(

. EJECT
```

```
LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

273.   'MODE'NFD='STRUCT('REF[,]'INT       TAB,                                  DEFN0100  >DEFN0100 B
274.                     'REF[]'INT TAB1,                                        DEFN0200            B(
275.                     'REF[,]'CHAR  VAL,                                      DEFN0300            B(
276.                     'REF[,]'INT RWTABADR,                                   DEFN0400            B(
277.                     'REF'FD FD,                                             DEFN0500            B(
278.                     'BOOL FD1,                                                                  B(
279.                     'INT CAF,                                               DEFN0600  >DEFN0600 B(
280.                     [1..6]'INT PO);                                         DEFN0700            B(
281.   # DEFINICE MODU NEFUNDAMENTALNI OBLASTI                  #                DEFN0800            B
282.   # INDEX  PREINDEXOVACI TABULKA                           #                DEFN0900            B
283.   # TAB    TABULKA SMERNIKU DO RWTABADR A DO REL           #                DEFN1000            B
284.   # TAB1   TABULKA SMERNIKU DO FD                          #                DEFN1100            B
285.   # VAL    TABULKA HODNOT ULOZENYCH V  NFD                 #                DEFN1200            B
286.   # FD     REFERENCE NA FD,KDE JSOU ULOZENY HODNOTY        #                DEFN1300            B
287.   # CAF    UKAZATEL NA VOLNOU ZONU V RWTABZA               #                DEFN1400            B
288.   # PO[1]  DELKA SETRIDENE ZONY                            #                DEFN1500            B
289.   # PO[2]  DELKA NESETRIDENE ZONY                          #                DEFN1600            B
290.   # PO[3]  POCET ZRUSENYCH HODNOT                          #                DEFN1700            B
291.   # PO[4]  C1 , PO[5]   C2  ,PO[6]   C3                    #                DEFN1800            B
292.  'PROC INSERT FD=('REF'FD FD,'STRING VAL,'REF'INT IN,'REF'BOOL BOOL)                            B
293.                                                  'VOID:                                       B
294.          (IN:=1);                                                                            B
295.    'PROC  TISK NFD =('NFD NFD)'VOID:                                       TISN0100  >TISN0100 B
296.    #    PROCEDURA PRO TISK NEFUNDAMENTALNI OBLASTI  #                      TISN0200            B
297.    #    VYTISKNE  SE TEZ  PRISLUSNA FD              #                      TISN0300            B
298.    (                                                                       TISN0400            B
299.         PRINT((NEWPAGE,"    C.RADKU                TAB      VAL NEBO TAB",TISN0500            B(
300.              NEWLINE))                                                     TISN0600            B(((
301.    ;    'FOR I'TO 'UPB TAB'OF NFD 'DO                                      TISN0700            B(
302.                   'IF # FD'OF NFD:=:'NIL #FD1 'OF NFD 'THEN                TISN0900            B(D
303.              PRINT((I,                    (TAB'OF NFD)[I,],               TISN0800  ?         B(DI
304.                                               (VAL'OF NFD)[I,],NEWLINE))  TISN0900            B(DI((
305.                                           'ELSE                                               B(DI
306.              PRINT((I,                    (TAB'OF NFD)[I,],               TISN0800  >TISN0800 B(DI
307.                                            (TAB1'OF NFD)[I],NEWLINE))     TISN1000            B(DI((
308.                                           'FI                                                 B(DI
309.         'OD                                                               TISN1200  >TISN1200 B(D
310.    ;    'IF # RWTABADR'OF NFD'ISNT 'NIL # (PO'OF NFD)[5] 'NE 0 'THEN      TISN1300            B(
311.             PRINT((NEWLINE,"    RWTABADR   ",NEWLINE))                    TISN1400            B(I
312.        ;     'FOR I'TO 'UPB  RWTABADR 'OF NFD 'DO                          TISN1500            B(I
313.                    PRINT(((RWTABADR'OF NFD)[I,],NEWLINE))                 TISN1600            B(ID
314.                'OD                                                        TISN1700            B(ID
315.           'FI                                                             TISN1800            B(I
316.         ;  'IF # FD'OF NFD'ISNT 'NIL #                                    TISN1900            B(
317.              'NOT FD1 'OF NFD 'THEN TISK FD(FD'OF NFD)                                         B(I
318.           'FI                                                             TISN2000  >TISN2000 B(I
319.    );                                                                     TISN2100            B(
320.   'PROC ADRINTO=([]'INT A,'REF[,]'INT RWTABADR,'REF'INT ADR,                                   B
321.                            'REF'INT CAF)'VOID:                                                B(
322.    # PROCEDURA ZAJISTI PREPSANI ADRES ULOZENYCH V A DO RWTABADR          #                     B

LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP
```

```
LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

323.  (     'INT I:=CAF;                                                                        B
324.        'INT L=2'UPB RWTABADR-1;                                                            B(
325.        ADR:=CAF;                                                                           B(
326.        'FOR J'BY L'DO                                                                      B(
327.           'IF J+L-1'LE 'UPB A 'THEN                                                        B(D
328.                 RWTABADR[I,2..'UPB A-J+1];                                                 B(DI
329.                 CAF:=RWTABADR[I,1];                                                        B(DI
330.                 RWTABADR[I,1]:=NULL;'GOTO EXIT                                             B(DI
331.                              'ELSE                                                         B(DI
332.                 RWTABADR[I,2..L+1]:=A[J..J+L-1];                                           B(DI
333.                 I:=RWTABADR[I,1];                                                          B(DI
334.                 CAF:=I                                                                     B(DI
335.           'FI                                                                              B(DI
336.        'OD;                                                                                B(D
337.  EXIT:    'SKIP                                                                           B(
338.    );                                                                                    B(
339.  'PROC GET ALL=('REF[]'INT A,'REF[,]'INT RWTABADR, 'INT ADR)'VOID:                          B
340.  # PROCEDURA PREDA VSECHNY ADRESY UKAZUJICI DO REL                          #              B
341.  # JEJICHZ SEZNAM ZACINA NA ADRESE ADR V RWTABZA  DO POLE A                 #              B
342.  ( 'INT I:=ADR,L:=2'UPB RWTABADR-1 ;                                                      B
343.        'FOR J 'BY L'DO                                                                   B(
344.           'IF 'UPB A 'GT J+L-1'THEN                                                      B(D
345.                 A[J..J+L-1]:=RWTABADR[I,2..L+1]                                          B(DI
346.                              'ELSE                                                       B(DI
347.                 A[J..'UPB A]:=RWTABADR[I,2..'UPB A-J+1];                                 B(DI
348.                 'GOTO EXIT                                                               B(DI
349.           'FI;                                                                           B(DI
350.           I:=RWTABADR[I,1]                                                               B(D
351.        'OD ;                                                                             B(D
352.  EXIT:    'SKIP                                                                           B(
353.  );                                                                                      B(
354.  'PROC GETALL1=('REF[]'INT A,'REF[,]'INT RWTABADR,'INT ADR,'REF'INT L )                   B
355.                                                                        'VOID:            B
356.  # PROCEDURA PREDA VSECHNY ADRESY UKAZUJICI DO REL V POLI A BEZ ZRUSENY#                  B
357.  # SMERNIKU,ZACATEK SEZNAMU JE V RWTABADR NA RADCE ADR                      #              B
358.  ( [1..'UPB A]'INT AO;                                                                   B
359.        GET ALL(AO,RWTABADR,ADR);                                                         B(
360.        L:=0;                                                                             B(
361.        'FOR I'TO 'UPB A'DO                                                               B(
362.           'IF AO[I]'NE NULL'THEN                                                         B(D
363.                 L+:=1 ;                                                                  B(DI
364.                 A[L]:=AO[I]                                                              B(DI
365.           'FI                                                                            B(DI
366.        'OD                                                                               B(D
367.  # PROCEDURA  VYZADUJE PROCEDURU GET ALL                                    #              B(
368.  );                                                                                      B(
369.  'PROC RETURN =('REF'INT ADR,'REF[,]'INT RWTABADR,'REF'INT CAFT)'VOID:                   B
370.  # UVOLNENE SEZNAMY SE VRACI SYSTEMU                                       #              B
371.  (     'INT I:=ADR ;                                                                     B
372.        RWTABADR[CAFT,1]:=ADR;                                                            B(

LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP
```

```
LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

373.   ON:   'IF RWTABADR[I,1]'NE NULL 'THEN                                                    B(
374.             I:=RWTABADR[I,1];                                                             B(I
375.             'GOTO ON                                                                      B(I
376.         'FI;                                                                              B(I
377.         CAFT:=I;                                                                          B(
378.         ADR:=-1                                                                           B(
379.   );                                                                                      B(
380.  'PROC SORT NFD=('REF'NFD NFD)'VOID:                                                      B
381.   # PROCEDURA SETRIDI NEFUNDAMENTALNI OBLAST  #                                           B
382.   # VYELI MINUJE PRAZDNA MISTA  Z RWTABADR    #                                           B
383.   # JESTLIZE EXISTUJI                         #                                           B
384.   (     'STRING VAL1,VAL2,VZ; 'INT Z ;'BOOL BOOL;                                        B
385.         BOOL:= FD'OF NFD:=:'NIL;                                                          B(
386.         'FOR I 'TO (PO'OF NFD)[2]-1'DO                                                    B(
387.            'FOR J'FROM I+1'TO(PO'OF NFD)[2]-1'DO                                          B(D
388.               'IF BOOL 'THEN                                                              B(DD
389.                     'IF(VAL'OF NFD)[(TAB'OF NFD)[I,3],] 'GT                               B(DDI
390.                       (VAL'OF NFD)[(TAB'OF NFD)[J,3],]'THEN                               B(DDII
391.                             Z:=(TAB'OF NFD)[I,3];                                         B(DDII
392.                            (TAB'OF NFD)[I,3]:=(TAB'OF NFD)[J,3];                          B(DDII
393.                            (TAB'OF NFD)[J,3]:= Z                                          B(DDII
394.                     'FI                                                                   B(DDII
395.                        'ELSE                                                              B(DDI
396.                            GET   (FD'OF NFD,VAL1,(TAB1'OF NFD)                            B(DDI
397.                                    [(TAB'OF NFD)[I,3]]);                                  B(DDI(
398.                           GET   (FD'OF NFD,VAL2,(TAB1'OF NFD)                             B(DDI
399.                                    [(TAB'OF NFD)[J,3]]);                                  B(DDI(
400.                           'IF VAL1>VAL2'THEN                                              B(DDI
401.                               Z:=(TAB'OF NFD)[I,3];                                       B(DDII
402.                               (TAB'OF NFD)[I,3]:=(TAB'OF NFD)[J,3];                       B(DDII
403.                               (TAB'OF NFD)[I,3]:=Z                                        B(DDII
404.                           'FI                                                             B(DDII
405.                  'FI                                                                      B(DDI
406.    # TABULKA TAB JE SETRIDENA         #                                                  B(DD
407.    # NYNI CISTTENI OD PRAZDNYCH DER   #                                                  B(DD
408.              'OD                                                                         B(DD
409.         'OD;                                                                             B(D
410.         'IF (PO'OF NFD)[5]'NE 0 'THEN                                                    B(
411.            #  BYL VYGENEROVANY PROSTOR PRO RWTABADR #                                    B(I
412.            'INT L,L1,ADR;                                                               B(I
413.            'FOR I'TO (PO'OF NFD)[2]'DO                                                  B(I
414.                L:=(TAB'OF NFD)[(TAB'OF NFD)[I,3],2];                                    B(ID
415.                'IF  L 'NE 0 'THEN                                                       B(ID
416.                    [1..L]'INT  A;                                                       B(IDI
417.                           GET ALL1(A,RWTABADR'OF NFD,ADR,L1);                           B(IDI
418.                           RETURN(ADR,RWTABADR'OF NFD,CAF 'OF NFD);                      B(IDI
419.                           ADR INTO(A[1..L1],RWTABADR'OF NFD,ADR,                        B(IDI
420.                                                        CAF'OF NFD);                     B(IDI(
421.                           (TAB'OF NFD)[(TAB'OF NFD)[I,3],2]:=L1;                        B(IDI
422.                           (TAB'OF NFD)[(TAB'OF NFD)[I,3],1]:=ADR                        B(IDI

LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP
```

```
LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

423.                'FI                                                                          B(IDI
424.             'OD                                                                             B(ID
425.          'FI;                                                                               B(I
426.          (PO'OF NFD)[1]:=(PO'OF NFD)[2]-:=(PO'OF NFD)[3];                                   B(
427.          (PO'OF NFD)[3]:=1                                                                  B(
428.    );                                                                                       B(
429.    'PROC ADD SPACE NFD=('REF 'NFD NFD)'VOID:                                                B
430.    ('SKIP);                                                                                 B
431.    'PROC FINDNFDINT    =('REF'NFD NFD,'INT    VALUE,'REF'INT IN,                            B
432.                          'REF'BOOL BOOL)'VOID:      ('SKIP );                               B
433.    'PROC FINDNFDSTRING =('REF'NFD NFD,'STRING VALUE,'REF'INT IN,                            B
434.                          'REF'BOOL BOOL)'VOID:                                              B(
435.     #  IN  UKAZUJE DO TAB[3,] #                                                            B
436.     #  VALUE  HODNOTA,JEJIZ REFERENCE SE HLEDA#                                            B
437.     (    'INT P,H,L,IO,I;                                                                  B
438.          P:=(PO'OF NFD)[1]-1;                                                              B(
439.          H:=P;                                                                             B(
440.          L:=1;                                                                             B(
441.          IO:=-1;                                                                           B(
442.          I:=1;                                                                             B(
443.    CYCLE:'IF I'NE IO'THEN                                                                  B(
444.             'IF (VAL 'OF NFD)[(TAB'OF NFD)[I,3],]=VALUE 'THEN                              B(I
445.                          BOOL:='TRUE;                                                      B(II
446.                          IN:=I;                                                            B(II
447.                          'GOTO FINDV                                                       B(II
448.             'FI;                                                                           B(II
449.             'IF ( VAL'OF NFD)[(TAB'OF NFD)[I,3],]<VALUE                                    B(I
450.                       'THEN  L:=I                                                          B(II
451.                       'ELSE H:=I                                                           B(II
452.             'FI;                                                                           B(II
453.             IO:=I;                                                                         B(I
454.             I:='ENTIER((L+H)/2);                                                           B(I
455.             'GOTO CYCLE                                                                    B(I
456.                                  'FI ;                                                     B(I
457.    #  HODNOTA NEBYLA NALEZENA V SETRIDENE ZONE #                                          B(
458.          'IF(PO'OF NFD)[2]'NE (PO'OF NFD)[1]'THEN                                        B(
459.             'FOR J'FROM (PO'OF NFD)[1]'TO(PO'OF NFD)[2]'DO                               B(I
460.                'IF( VAL'OF NFD)[(TAB'OF NFD)[J,3],]=VALUE 'THEN                          B(ID
461.                      BOOL:='TRUE;                                                        B(IDI
462.                      I:=J;                                                               B(IDI
463.                      'GOTO FINDV                                                         B(IDI
464.                'FI                                                                       B(IDI
465.             'OD                                                                          B(ID
466.          'FI;                                                                            B(I
467.    #  HODNOTA NENI ULOZENA V NFD  #                                                       B(
468.      I:=-999;                                                                            B(
469.       BOOL:='FALSE;                                                                      B(
470.       FINDV:                                                                             B(
471.          IN:=I                                                                           B(
472.    );                                                                                    B(

LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP
```

```
LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

473.   'PROC FIND NFD =('REF'NFD NFD,'STRING VAL,'REF 'INT I,'REF'BOOL BOOL)                    B
474.                                                          'VOID:                           B
475.   #  PROCEDURA ZAJISTI NALEZENI ODKAZU PODLE DANE HODNOTY  #                               B
476.   (                                                                                       B
477.         'IF FD'OF NFD:=:'NIL 'THEN                                                        B(
478.                              FINDNFDSTRING(NFD,VAL,I,BOOL)                                B(I
479.                             #HODNOTA ULOZENA PRIMO V NFD  #                               B(I
480.                            'ELSE                                                          B(I
481.                             #HODNOTA ULOZENA VE FD #                                      B(I
482.                               FIND   (FD'OF NFD,VAL,BOOL,I);                              B(I
483.                               'IF  BOOL'THEN                                              B(I
484.                                    'INT IN;                                               B(II
485.                                    IN:=I;                                                 B(II
486.                                    FINDNFDINT(NFD,IN,I,BOOL)                              B(II
487.                     'FI                                                                   B(II
488.         'FI                                                                               B(I
489.   );                                                                                      B(
490.   'PROC INSERT NFD=('REF'NFD NFD,'INT ADR,'STRING VAL,'REF'INT INADR)                      B
491.                                                                 'VOID:                    B
492.   #PROCEDURA ZALOZI RETEZ VAL DO NFD A DO RWTABADR ULOZI NA PRISLUSNE  #                   B
493.   #MISTO HODNOTU SMERNIKU ADR,KTERY JE CISLEM RADKU   ODPOVIDAJICI N-TIC#                  B
494.   (     'INT J,I,L,K;                                                                     B
495.          'BOOL BOOL; 'INT IN;                                                             B(
496.         'IF # FD'OF NFD :=:'NIL # FD1 'OF NFD 'THEN                                       B(
497.          FIND NFD STRING(NFD,VAL,I,BOOL);                                                 B(I
498.         'IF 'NOT BOOL'THEN                                                                B(I
499.          # VKLADANI DO SLOVNIKU #                                                         B(II
500.             'IF(PO'OF NFD)[2]'GT(PO'OF NFD)[4] 'THEN                                     B(II
501.                              'IF (PO'OF NFD)[3]>1'THEN                                   B(III
502.                                    SORT NFD(NFD)                                         B(IIII
503.                                         'ELSE                                            B(IIII
504.                                    ADD SPACE NFD(NFD)                                    B(IIII
505.                         'FI                                                              B(IIII
506.             'FI;                                                                         B(III
507.          I:=(PO'OF NFD)[2];                                                              B(II
508.          (VAL'OF NFD)[(TAB'OF NFD)[I,3],]:=VAL;                                          B(II
509.          (PO'OF NFD)[2]+:=1                                                              B(II
510.        'FI                                                                               B(II
511.          'ELSE                                                                           B(I
512.          INSERT   (FD'OF NFD,VAL,IN,BOOL);                                               B(I
513.          FIND NFD INT(NFD, IN,I,BOOL);                                                   B(I
514.          'IF'NOT BOOL'THEN                                                               B(I
515.             # NEBYLO V NFD  #                                                           B(II
516.               'IF (PO'OF NFD)[2]'GT (PO'OF NFD)[4]'THEN                                 B(II
517.                    'IF(PO'OF NFD)[3] > 1'THEN                                           B(III
518.                                SORT NFD(NFD)                                            B(IIII
519.                                     'ELSE                                               B(IIII
520.                                ADD SPACE NFD(NFD)                                       B(IIII
521.                    'FI;                                                                 B(IIII
522.                    I:=(PO'OF NFD)[2];                                                   B(III

LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP
```

```
LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

523.                              (TAB1'OF NFD)[(TAB'OF NFD)[I,3]]:=IN;                          B(III
524.                              (PO'OF NFD)[2]+:=1                                             B(III
525.                         'FI                                                                 B(III
526.                    'FI                                                                      B(II
527.               'FI;                                                                          B(I
528.               INADR:=(TAB'OF NFD)[I,3];                                                     B(
529.               #  POLOZKA JE ZALOZENA  A HODNOTA UKLADANA DO REL JE URCENA#                  B(
530.               # NYNI JE NUTNE ZAPSAT ADR DO RWTABADR                          #            B(
531.               J:=(TAB'OF NFD)[(TAB'OF NFD)[I,3],1];                                         B(
532.               L:=(TAB'OF NFD)[(TAB'OF NFD)[I,3],2];                                         B(
533.               K:=2'UPB RWTABADR 'OF NFD-1;                                                  B(
534.               'WHILE (RWTABADR'OF NFD)[J,1]'NE NULL 'DO                                     B(
535.                  L-:=K;                                                                     B(D
536.                  J:=(RWTABADR'OF NFD)[J,1]                                                  B(D
537.               'OD;                                                                          B(D
538.               'IF L'GE K 'THEN                                                              B(
539.                   # ZALOZIT NOVOU RADKU DO SEZNAMU #                                       B(I
540.                  'IF(RWTABADR 'OF NFD)[CAF 'OF NFD,1]=NULL 'THEN                           B(I
541.                       # NENI  VOLNA RADKA V RWTABADR #                                     B(II
542.                       # ADD SPACE NFD RWTABADR(NFD)  #                                     B(II
543.                       'SKIP                                                               B(II
544.                   'FI;                                                                    B(II
545.                   J:=(RWTABADR'OF NFD)[J,1]:=CAF'OF NFD;                                  B(I
546.                   CAF'OF NFD:=(RWTABADR'OF NFD)[CAF'OF NFD,1];                            B(I
547.                   (RWTABADR'OF NFD)[J,1]:=NULL;                                           B(I
548.                   (RWTABADR'OF NFD)[J,2]:=ADR                                             B(I
549.                   'ELSE                                                                   B(I
550.                     # V RADKU JE MISTO #                                                   B(I
551.                     (RWTABADR'OF NFD)[J,L+1+1]:=ADR                                       B(I
552.                   'FI;                                                                    B(I
553.                  (TAB'OF NFD)[I,2]+:=1                                                    B(
554.                  # ZVETSENI DELKY O 1 #                                                   B(
555.    );                                                                                     B(
```

. EJECT

```
  LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

  556.       'FD FD                                                                              B
  557.   ;  'INT C1:=50,C2:=10,C3,C4                                                            B
  558.    ; PRINT(("VYPOCET",NEWLINE))                                                          B
  559.   ; REF'OF FD:='LOC    [1..C1]'INT                                                       B
  560.   ;   VAL'OF FD:='LOC [1..C1,1..C2]'CHAR                                                 B
  561.    ;                                                                                     B
  562.        'FOR J 'TO  C1 'DO                                                                B
  563.        (REF 'OF FD)[J]:=J;                                                               BD
  564.             'FOR I'TO C2'DO                                                              BD
  565.         (VAL 'OF FD)[J,I]:="Z"                                                           BDD
  566.             'OD                                                                          BDD
  567.        'OD                                                                               BD
  568.    ;   PO'OF FD:=(1,1,1,1)                                                              B
  569.    ;  TISK FD(FD)                                                                       B
  570.    ;                                                          ININ1200  >ININ1200       B

. EJECT
```

```
LINE  ↑1..:...1↑0...:...2↑0...:...3↑0...:...4↑0...:...5↑0...:...6↑0...:...7↑0...:...8↑0          NESTING MAP

571.    'NFD  NFD;                                                                              B
572.  C1:=100;   C2:=50;    C3:=20;    C4:=7;                                                   B
573.   # PROCEDURA ZAJISTI VYGENEROVANI PRISLUSNEHO PAMETOVEHO    #          ININ0200  >ININ0200 B
574.   # PROSTORU NUTNEHO K ULOZENI NEFUNDAMENTALNI OBLASTI       #          ININ0300            B
575.   # VSTUPNI PARAMETRY                                        #          ININ0400            B
576.   # NFD  PROMENNA MODU NFD                                   #          ININ0500            B
577.   # C1   KARDINALITA TABULEK INDEX,TAB                       #          ININ0600            B
578.   # C2   KARDINALITA TABULEK RWTABADR                        #          ININ0700            B
579.   # C4   POCET ZNAKU VE VAL[I,]                              #                              B
580.   # C3   POCET SLOUPCU TABULKY RWTABADR                      #          ININ0800  >ININ0800 B
581.   # FD   REFERENCE NA PRISLUSNOU FUNDAMENTALNI OBLAST        #          ININ0900            B
582.   # JE-LI C2 =0  PAK TAB[1:C1]                               #          ININ1000            B
583.   # JE-LI FD=NIL PAK SE INICIALIZUJE VAL MISTO TAB1          #          ININ1100            B
584.     'IF C2 =0 'THEN                                                     ININ1300            B
585.                    RWTABADR'OF NFD:='NIL                                ININ1400            BI
586.               'ELSE                                                     ININ1500            BI
587.                     RWTABADR'OF NFD:='LOC [1..C2,1..C3]'INT             ININ1600            BI
588.     'FI;                                                                ININ1700            BI
589.           TAB'OF NFD:='LOC  [1..C1,1..3]'INT                            INIT1750            B
590.    ; 'IF #FD:=:'NIL# 'FALSE 'THEN                                                          B
591.        VAL 'OF NFD:='LOC [1..C1,1..C4]'CHAR                                                BI
592.     'ELSE                                                                                  BI
593.        TAB1 'OF NFD:='LOC[1..C1]'INT                                                       BI
594.     'FI                                                                                    BI
595.     # PROSTOR VYGENEROVAN      #                                      ININ2300  >ININ2300 B
596.     ;  FD'OF NFD:=FD                                                  ININ2400            B
597.     ;  PO'OF NFD:=(1,1,1,C1,C2,C3)                                    ININ2500            B
598.     #   VSEM PROMENNYM JSOU PRIRAZENY HODNOTY    #                   ININ3700            B
599.     ; 'FOR I 'TO C1 'DO                                               ININ2600  ?         B
600.            (TAB'OF NFD)[I, ]:=(0,0,I)                                 ININ2900            BD
601.        ; 'IF # FD:=:'NIL # 'FALSE 'THEN                                                    BD
602.            FD1 'OF NFD:=#FD:=:'NIL#'FALSE;                                                 BDI
603.            'FOR J 'TO C4 'DO                                                               BDI
604.               (VAL 'OF NFD)[I,J]:="Z"                                                      BDID
605.            'OD                                                                             BDID
606.                                 'ELSE                                 ININ3300  >ININ3300 BDI
607.               (TAB1'OF NFD)[I]:=-1                                    ININ3400            BDI
608.          'FI                                                          ININ3500            BDI
609.       'OD                                                             ININ3600            BD
610.      ; 'FOR I     'TO C2-1 'DO                                                             B
611.            (RWTABADR 'OF NFD)[I,1]:=I+1                                                    BD
612.       'OD;                                                                                 BD
613.            (RWTABADR 'OF NFD)[C2,1]:=NULL                                                  B
614.  ;                                                                                         B
615.  #                                                                                         B
616.  #                                                                                         B
617.    TISK NFD(NFD)                                                                           B
618.   'END                                                                                     B

. FILESIN *1
```

```
. LOAD NASE,NENT
```

```
MESSAGE   ORIGIN P4                        I  400,LENGTH OF MODUL A68      H:41DC D:16860
**********************    JOB NAME : SKALA          EO :          DATE : 18/11/78-322    TIME : 15.30.28    **********************
*                                                                                                                                    *
*  END OF STEP  *    STEP NO : 002    ELAPSED TIME : 0003.08                                                                         *
*                                                                                                                                    *
************************************************************************************************************************************

        . LOAD NASE,NENT

                                                                                                          /LOAD      76156

NASE,NENT                                                                                                                          *
```

LIST OF CHANNELS:	DV	AREA	RESET	SET	GET	PUT	BIN	COMPRS	MAXNF	MAXPAGE	MAXLINE	MAXCHAR
STANDINCHANNEL	CRQ0		NO	NO	YES	NO	YES	NO	1	1	65535	80
MT0CHANNF	MT 0		YES	NO	YES	YES	YES	NO	1	1	65535	65535
MT1CHANNF	MT 1		YES	NO	YES	YES	YES	NO	1	1	65535	65535
STANDOUTCHANNEL	PR 0		NO	NO	NO	YES	NO	NO	1	65535	65	160

LIST OF BOOKS:	FN	FC	FS	BS	BN	RS	RP	CD
S1	S1	1		80	2	80	000	00000
S3	S3	1		122	2	122	000	00000

VYPOCET

TISK HODNOT FUNDAMENTALNI OBLASTI

POLE PO

+1	+1	+1	+1
C.RADKU	REF	VAL	
+1	+1	ZZZZZZZZZZ	ZZZZZZZZZZ
+2	+2	ZZZZZZZZZZ	ZZZZZZZZZZ
+3	+3	ZZZZZZZZZZ	ZZZZZZZZZZ
+4	+4	ZZZZZZZZZZ	ZZZZZZZZZZ
+5	+5	ZZZZZZZZZZ	ZZZZZZZZZZ
+6	+6	ZZZZZZZZZZ	ZZZZZZZZZZ
+7	+7	ZZZZZZZZZZ	ZZZZZZZZZZ
+8	+8	ZZZZZZZZZZ	ZZZZZZZZZZ
+9	+9	ZZZZZZZZZZ	ZZZZZZZZZZ
+10	+10	ZZZZZZZZZZ	ZZZZZZZZZZ
+11	+11	ZZZZZZZZZZ	ZZZZZZZZZZ
+12	+12	ZZZZZZZZZZ	ZZZZZZZZZZ
+13	+13	ZZZZZZZZZZ	ZZZZZZZZZZ
+14	+14	ZZZZZZZZZZ	ZZZZZZZZZZ
+15	+15	ZZZZZZZZZZ	ZZZZZZZZZZ
+16	+16	ZZZZZZZZZZ	ZZZZZZZZZZ
+17	+17	ZZZZZZZZZZ	ZZZZZZZZZZ
+18	+18	ZZZZZZZZZZ	ZZZZZZZZZZ
+19	+19	ZZZZZZZZZZ	ZZZZZZZZZZ
+20	+20	ZZZZZZZZZZ	ZZZZZZZZZZ
+21	+21	ZZZZZZZZZZ	ZZZZZZZZZZ
+22	+22	ZZZZZZZZZZ	ZZZZZZZZZZ
+23	+23	ZZZZZZZZZZ	ZZZZZZZZZZ
+24	+24	ZZZZZZZZZZ	ZZZZZZZZZZ
+25	+25	ZZZZZZZZZZ	ZZZZZZZZZZ
+26	+26	ZZZZZZZZZZ	ZZZZZZZZZZ
+27	+27	ZZZZZZZZZZ	ZZZZZZZZZZ
+28	+28	ZZZZZZZZZZ	ZZZZZZZZZZ
+29	+29	ZZZZZZZZZZ	ZZZZZZZZZZ
+30	+30	ZZZZZZZZZZ	ZZZZZZZZZZ
+31	+31	ZZZZZZZZZZ	ZZZZZZZZZZ
+32	+32	ZZZZZZZZZZ	ZZZZZZZZZZ
+33	+33	ZZZZZZZZZZ	ZZZZZZZZZZ
+34	+34	ZZZZZZZZZZ	ZZZZZZZZZZ
+35	+35	ZZZZZZZZZZ	ZZZZZZZZZZ
+36	+36	ZZZZZZZZZZ	ZZZZZZZZZZ
+37	+37	ZZZZZZZZZZ	ZZZZZZZZZZ
+38	+38	ZZZZZZZZZZ	ZZZZZZZZZZ
+39	+39	ZZZZZZZZZZ	ZZZZZZZZZZ
+40	+40	ZZZZZZZZZZ	ZZZZZZZZZZ
+41	+41	ZZZZZZZZZZ	ZZZZZZZZZZ
+42	+42	ZZZZZZZZZZ	ZZZZZZZZZZ
+43	+43	ZZZZZZZZZZ	ZZZZZZZZZZ
+44	+44	ZZZZZZZZZZ	ZZZZZZZZZZ
+45	+45	ZZZZZZZZZZ	ZZZZZZZZZZ
+46	+46	ZZZZZZZZZZ	ZZZZZZZZZZ
+47	+47	ZZZZZZZZZZ	ZZZZZZZZZZ
+48	+48	ZZZZZZZZZZ	ZZZZZZZZZZ
+49	+49	ZZZZZZZZZZ	ZZZZZZZZZZ
+50	+50	ZZZZZZZZZZ	ZZZZZZZZZZ

C.RADKU	TAB	VAL NEBO TAB	
+1	+0	+0	+1 -1
+2	+0	+0	+2 -1
+3	+0	+0	+3 -1
+4	+0	+0	+4 -1
+5	+0	+0	+5 -1
+6	+0	+0	+6 -1
+7	+0	+0	+7 -1
+8	+0	+0	+8 -1
+9	+0	+0	+9 -1
+10	+0	+0	+10 -1
+11	+0	+0	+11 -1
+12	+0	+0	+12 -1
+13	+0	+0	+13 -1
+14	+0	+0	+14 -1
+15	+0	+0	+15 -1
+16	+0	+0	+16 -1
+17	+0	+0	+17 -1
+18	+0	+0	+18 -1
+19	+0	+0	+19 -1
+20	+0	+0	+20 -1
+21	+0	+0	+21 -1
+22	+0	+0	+22 -1
+23	+0	+0	+23 -1
+24	+0	+0	+24 -1
+25	+0	+0	+25 -1
+26	+0	+0	+26 -1
+27	+0	+0	+27 -1
+28	+0	+0	+28 -1
+29	+0	+0	+29 -1
+30	+0	+0	+30 -1
+31	+0	+0	+31 -1
+32	+0	+0	+32 -1
+33	+0	+0	+33 -1
+34	+0	+0	+34 -1
+35	+0	+0	+35 -1
+36	+0	+0	+36 -1
+37	+0	+0	+37 -1
+38	+0	+0	+38 -1
+39	+0	+0	+39 -1
+40	+0	+0	+40 -1
+41	+0	+0	+41 -1
+42	+0	+0	+42 -1
+43	+0	+0	+43 -1
+44	+0	+0	+44 -1
+45	+0	+0	+45 -1
+46	+0	+0	+46 -1
+47	+0	+0	+47 -1
+48	+0	+0	+48 -1
+49	+0	+0	+49 -1
+50	+0	+0	+50 -1
+51	+0	+0	+51 -1
+52	+0	+0	+52 -1
+53	+0	+0	+53 -1
+54	+0	+0	+54 -1
+55	+0	+0	+55 -1
+56	+0	+0	+56 -1
+57	+0	+0	+57 -1
+58	+0	+0	+58 -1
+59	+0	+0	+59 -1
+60	+0	+0	+60 -1
+61	+0	+0	+61 -1
+62	+0	+0	+62 -1
+63	+0	+0	+63 -1

+65	+0	+0	+65	-1
+66	+0	+0	+66	-1
+67	+0	+0	+67	-1
+68	+0	+0	+68	-1
+69	+0	+0	+69	-1
+70	+0	+0	+70	-1
+71	+0	+0	+71	-1
+72	+0	+0	+72	-1
+73	+0	+0	+73	-1
+74	+0	+0	+74	-1
+75	+0	+0	+75	-1
+76	+0	+0	+76	-1
+77	+0	+0	+77	-1
+78	+0	+0	+78	-1
+79	+0	+0	+79	-1
+80	+0	+0	+80	-1
+81	+0	+0	+81	-1
+82	+0	+0	+82	-1
+83	+0	+0	+83	-1
+84	+0	+0	+84	-1
+85	+0	+0	+85	-1
+86	+0	+0	+86	-1
+87	+0	+0	+87	-1
+88	+0	+0	+88	-1
+89	+0	+0	+89	-1
+90	+0	+0	+90	-1
+91	+0	+0	+91	-1
+92	+0	+0	+92	-1
+93	+0	+0	+93	-1
+94	+0	+0	+94	-1
+95	+0	+0	+95	-1
+96	+0	+0	+96	-1
+97	+0	+0	+97	-1
+98	+0	+0	+98	-1
+99	+0	+0	+99	-1
+100	+0	+0	+100	-1

RWTABADR

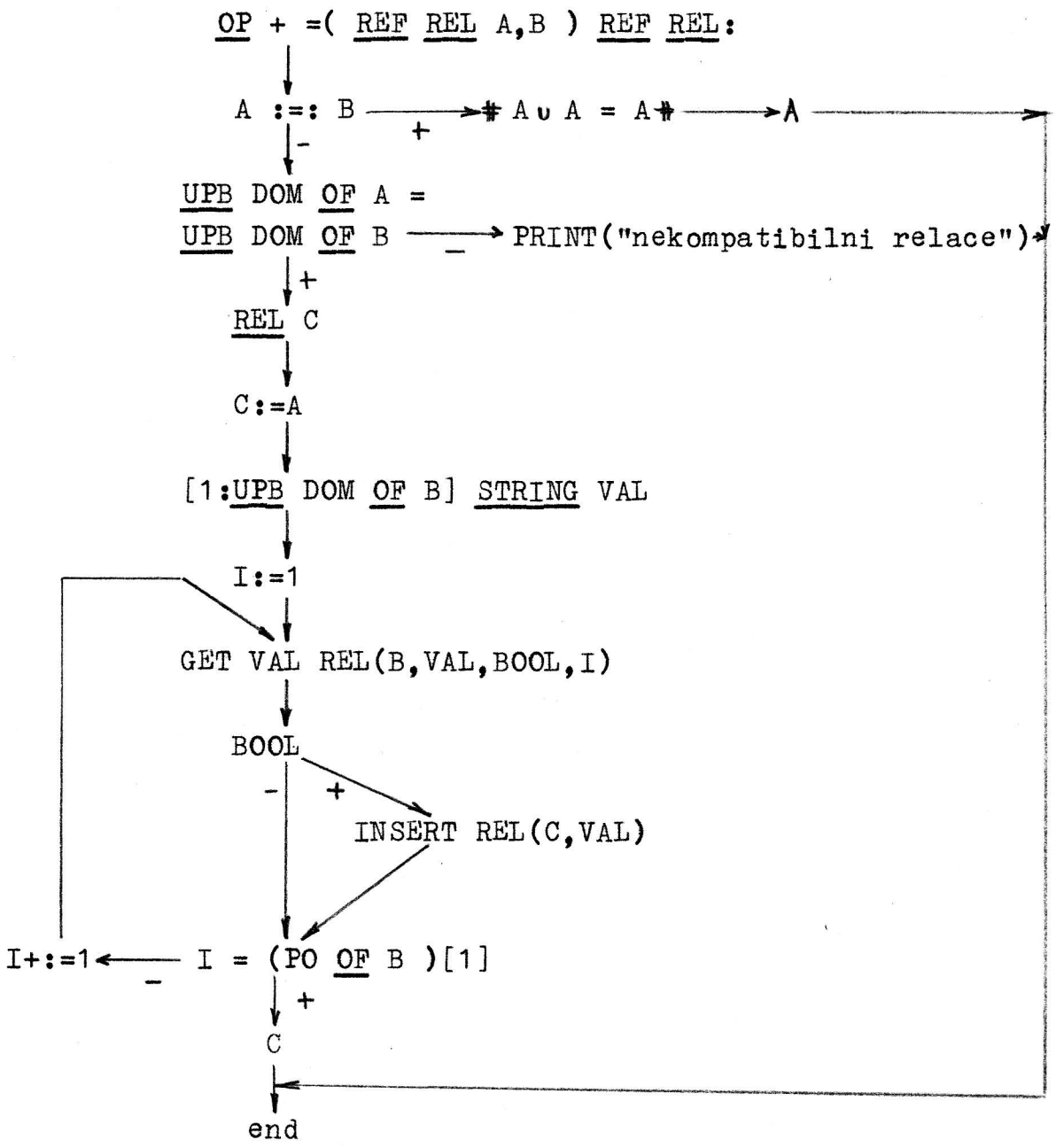
+2	+4	-880897288	+33685504	+302023928	+8585216	+65560	+1	+50	+10
+1	+10	+1	-880896592	+33619968	+33256	+8509442	+262156	+1	+4
+3	+7	+50	+1	+0	+33587916	-880896628	+65536	+301990304	+285696
+328716	+1	+5	+1	-880896628	+33619968	+302023928	+8710656	+65804	+1
+4	+1	-880896628	+65536	+301990336	+286976	+590860	+1	+9	+1
-880896628	+33619968	+302023928	+8710656	+65804	+1	+10	+1	+21666	-16798788
+5	-925926998	+34056	+34376	-704648236	-50348676	+33577364	+1.2	+32240	+36336
+184583432	+29204	+0	+0	+33577364	+33909	+0	+0	-1879048224	+369115780
+6	+24	-100695536	+36336	+184583432	+29204	+0	+0	+33559630	+5194
+33588792	+34736	+65560	+1	+17390	-100695536	+36336	+184583432	+29204	+34999
+7	+33577232	+5194	+33588792	+22932	+34376	+34872	+369120418	-1275085490	+369117654
+50	-880897288	+11	+0	+0	+721031	+2014888462	-4173829	+356560916	-926975796
+8	+1107382296	-806622223	-806631937	-806618897	+235998737	-822120284	+234996225	-5636096	-1107329113
-806721024	-5847044	+1342313220	+1073795586	-6783184	-938772342	+50	-13488642	+538976288	+538976288
+9	-13488642	-7569616	-936675222	-1375895667	-13488642	-7962832	-935626662	-1375895673	-13488642
-8356048	-934578102	-1879068926	+369115780	+33554792	+24	-100695536	+36336	+184583432	+29204
+10	+0	+33559630	+5194	+33588792	+22932	+34376	+34736	+1	+4
+1	+7	+21666	-16798788	+22176	+21666	-16798788	+22176	+274944	+459788
+11	+7	+1	-144315679	-805353248	-144332034	+34376	+34872	+369120418	-16798788
+21134	+34376	+32240	+36336	+184583432	+29204	+0	+0	+22932	+34736
+12	+34872	-923830400	+1006850060	+822779909	-906004548	+352469015	-928024730	+939741196	+238238723
+1342978051	+1342978050	+1343163954	-1006862522	-939688119	+1006784524	+237222914	-12040195	+1183011	-926976202
+13	-330806400	-183647234	+1343175682	+1343372290	+1343598850	-12235011	-12432512	-199938564	-6062280
-1577222236	-12562691	-4034560	+2080421890	-150157326	+543044736	-1610921088	-196620555	-806950144	-805353296
+14	-64991491	+2143522816	+2080421890	-152778766	+543044736	-704664810	+1275091608	-704665058	+1275091608
+543044736	-1610920195	+2143522816	+2080421890	-155400206	+543044736	-1610889223	-805438727	-924879344	-1107394623

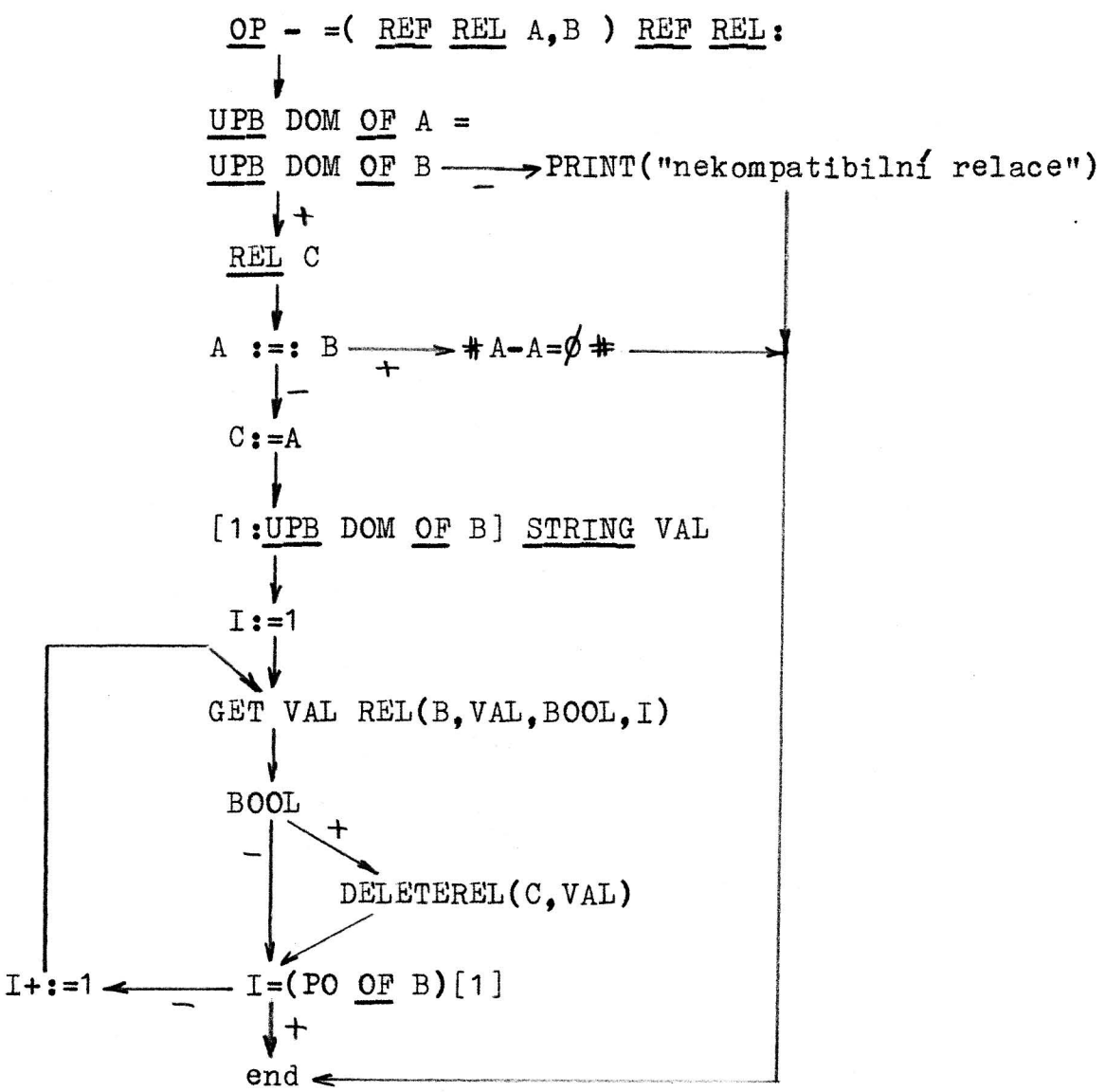
-12766722	-9138249	-1375895648	-13291009	-6320199	-1376026738	-13152514	-12235011	-12423680	-12239361
+16	-880839658	-1577222258	-12562690	-12239361	-12284003	-880839658	-1376026747	-13164034	-8093887
-822313021	-268467200	-939690461	+940449886	+880861188	-880839608	+822849541	+352366592	-928024914	-822166016
+17	-880834816	-268467200	-268588357	-940146499	-939556678	-822185776	+218216272	-170569993	+46592
-170659728	-352359768	-924879914	-1007261041	-939622583	+1127813120	+113665	-358878745	-1006731447	-906005546
+18	+0	+0	+0	+0	+0	+0	+0	+1509949765	+1381125970
+542002990	+542660696	+538987855	+1313428559	+1377849158	+541675081	+1128874316	+1230192980	+1229934138	+538976288
+19	+538976288	+538976288	+538976288	+538976288	+538976288	+538976288	+538976288	+538976288	+538976288
+538976288	+538976288	+538976288	+538976288	+538976288	+538976288	+538976288	+538976288	+538976288	+538976288
+20	-173061889	-189577220	+2135339265	-254807316	-924879692	-939884352	+256227168	-189575429	-4066900
-822378679	+223655744	-187478273	+7868241	-906005276	+308170752	-822313021	-268467200	-939690805	+940449886
+21	-880839608	-963676752	-1157663344	+536983360	-190628352	-175120385	+973152256	+122798897	-178135046
-1107462781	+537047299	-175214592	+2080421890	-192100366	+543044736	-1610928723	-178159679	-963676752	-356747888
+22	-880834816	-268467200	-268588357	-822316488	+302133251	+168276227	-231244289	+167906047	+905969664
-304119871	-1383497792	-4142592	+15106	+1885353985	+1342323201	+268504609	+973172834	-822280192	+83965728
+23	-199374336	+45316	+342018	+536901696	+805342246	+202506657	+94484288	-201590369	+17940225
-203913793	+805342247	+805568512	+18003744	-200834113	+805342244	-906005496	+23726593	+47105	-11993569
+24	+269531136	+1525004289	+1493385749	+268435456	-1375819939	+1491843072	+1520678913	+1502691861	+268435456
-880834816	-268458288	-268521296	-268458288	-939691109	+940449886	+880861188	+302006272	+304103425	+103070724
+25	+1500710688	-1056987435	+119586304	-211227168	+1073758838	+1073823744	-822316526	+302133251	+168276227
-233734657	+167949824	+96206851	+1279873876	+542066208	+1128808782	+1313164371	+975183904	+538976324	+1444945952
+26	+1161895968	+538976288	+1380275013	+1411404613	+1411391520	+1195725856	+538988629	+1411391520	+1112100384
+541282125	+1347572512	+538987841	+1481524768	+538976333	+1096306753	+1195712544	+1296128076	+1229866272	+541933912
+27	+538976288	+538976288	+216137	+1398022223	+1176519247	+1330336570	+541478432	+538976288	+538976288
+538976288	+538976288	+1178804256	+538976326	+1394614304	+538976288	+538976288	+1112744002	+1310728224	+542266144
+28	+542265376	+538976288	+541279264	+538976288	+538976288	+538976288	+538976288	+538976288	+538976288
+538976288	+538976288	+538976288	+538976256	+512	+455802696	+289492039	+1330389046	+941640014	+1229146433
+29	+1414090574	+0	+2108159	+1209156946	+1380930131	+541349202	+1229866784	+1229867348	+1229016137
+1396790345	+1330511872	+131081	+721373203	+1398030674	+1409323008	+47106	+2135339265	-254807314	+924844032
+30	+2135341057	+2135100705	+929038336	-822251312	-1006796614	+905969664	+65793	+16843009	+16843009
+16843009	+16843009	+16843009	+16843009	+16843009	+16843009	+16843009	+16843009	+16843009	+16843009
+31	+16843009	+16843009	+16777216	+0	+0	+0	+257	+16843009	+16843009
+0	+0	+0	+0	+0	+0	+0	+0	+0	+0
+32	+0	+0	+1	+16843009	+16843009	+16777216	+0	+16843009	+16843009
+16843009	+16843009	+16843009	+16843009	+16842752	+0	+0	+0	+0	+0
+33	+0	+0	+0	+65793	+16843009	+16843009	+16843009	+16843009	+16843009
+16843008	+0	+520093728	+538976288	+538976288	+538976288	+538976288	+538976288	+538976288	+538976288
+34	+0	+0	+1129447680	+1129316864	+1414660864	+1414530048	+1347552512	+1297352192	+1146423040
+538968319	+0	+45322	-254984704	-255008768	-302092082	-1007325009	-906004132	-354324274	-929075114
+35	-305968256	-183678986	-256968218	-1007325009	-973180722	+101826049	-254951936	-536944640	-906004132
-973180722	+87144288	-265728257	+250369	+1577881610	-822214657	-822316851	-14731776	-12201984	-496066060
+36	-12235314	-266647554	+2135299372	+536917792	-111457792	-113246218	-822444247	-1157664562	+1342289744
-273857025	-254936849	-1040289829	-1157664563	+1342551888	-273859072	-254889984	+302010369	-1006997335	+939872258
+37	+84260688	-111456514	-316454400	-275034369	+1642065	+352342038	-928026640	-822252158	-33592576
-257668944	-275952768	-183677196	-353473025	-257668608	-112116224	-255139840	-822447964	-1157730099	+1342224160
+38	+352342018	-928026760	-822252192	-906006628	-939757399	+1157648386	+806663968	-284706048	+1342320662
-924881092	-822252214	-928026724	+973230082	-939888471	+1040338964	-939888468	+939806722	+939606021	+83998496
+39	+268439574	+939606027	+83998496	-284705301	+268439574	-336564021	-939888471	+1375752213	-255178240
-111448577	+1115141632	-111448576	+1115141632	-111458309	-257180677	+1342984204	-302026586	-906004132	-302223194
+40	-906004132	-1107332916	+536916235	-64996609	-255045632	-924881254	+822849541	+352366592	-928026946
-33599746	-321762432	-183678464	-111425543	-805438727	-928027006	-33599746	-325104768	-183678464	-111458309
+41	-197538050	+1343011329	-257560097	+939597824	+219395888	-295582723	+536917504	-111449344	-257668944
-111456514	-327778816	-880839410	-906004142	-302026587	-906004132	-1107329211	+536920576	-12328960	-1543536827
+42	-12358113	-939888471	-1140883643	+1343731488	-111449344	-12367648	-301487616	+1343783099	-906004132
-402620603	+1409306648	-12301824	-111411200	+134283264	+520	+512	+16777216	+0	+0
+43	+0	+0	+65537	+0	+0	+0	+1095054640	+808209737	+1397967182
+1193296965	+1279872329	+1413829152	+740310304	+723537746	+539959375	+1377847630	+1129271890	+1162040352	+1128808782
+44	+1229210958	+1414088265	+1163010373	+892350764	+1296651091	+1346718789	+1142965064	+1095650885	+1277184324
+1162761289	+1179206994	+356857136	+841763150	+1129271890	+1162040352	+1145394761	+1128604997	+892351276	+1229865807
+45	+1411400773	+1447641925	+542068256	+1414023507	+541280321	+1313752396	+592917808	+875316040	+1095650885
+1277187406	+1431520580	+541675040	+1414023507	+542134863	+1196573005	+457389360	+892096594	+1330074177	+1293960536
+46	+1229866784	+909396802	+272839984	+908873033	+1397967182	+1193301588	+1446139701	+808922197	+1313558095
+1464737858	+1330596640	+675434319	+1260407108	+1163403341	+1096360002	+1159745358	+1280909395	+1311526990	+743198249
+47	+942429256	+1230184514	+1330596640	+1463898912	+1313821728	+1279872587	+1162092628	+1327517778	+1330074178

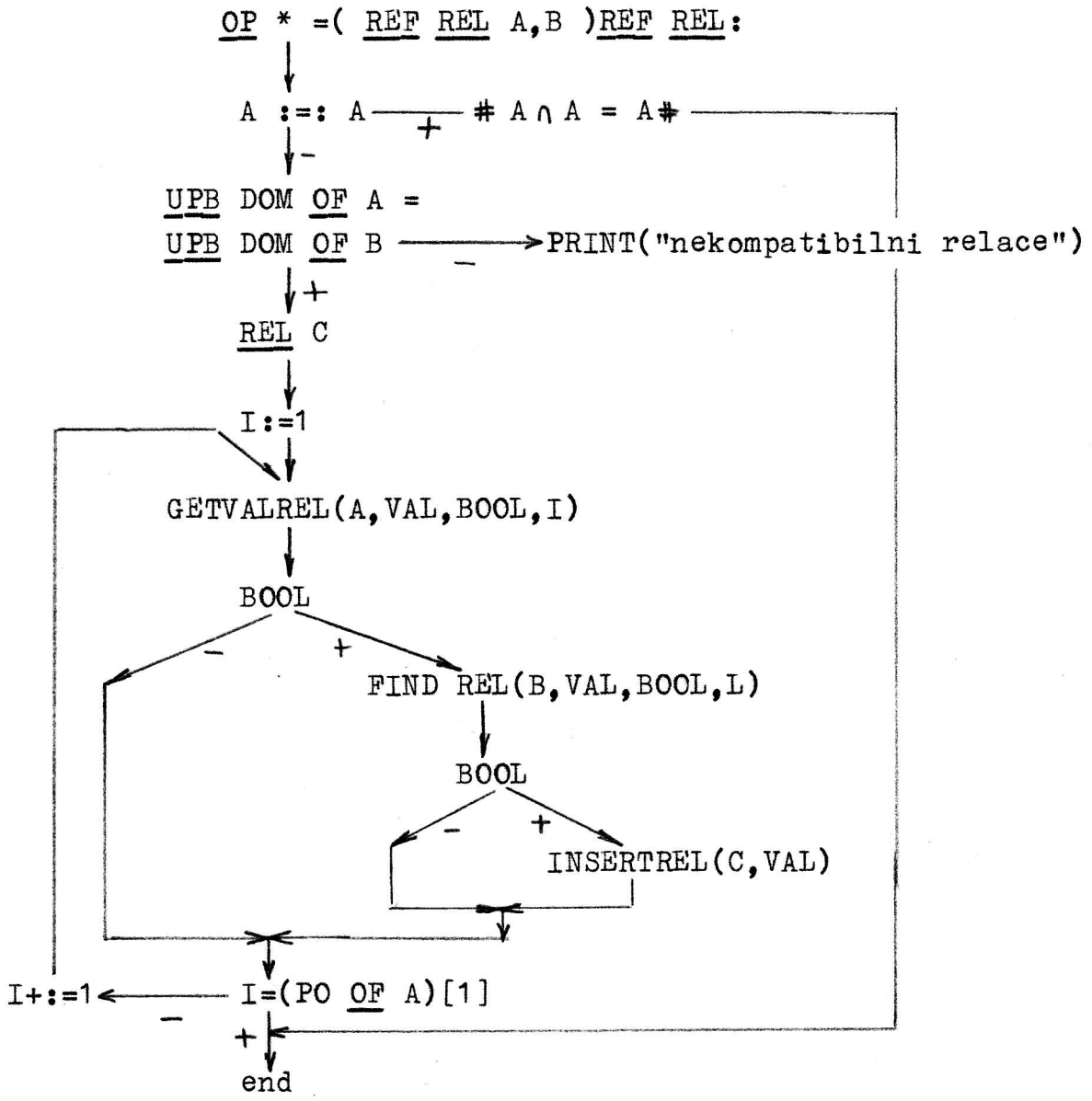
```
        +48 +1380930081 +1127559473  +743194947 +1260408646  +542331983 +1380009797  +541478738  +541280321 +1313752396
+1397031823   -43260307 -1632437135  -412373395 -1765605376    +9769363 -1902248847   -43323795 -2036466575  -412352915
        +49 -1217661331 -1499202447 -1217663232    +2032527 -1217734912    +5440399   -43272596  +559022991  -412373396
 +691863552    +9769364  +827917201  +177486228 +1095631761  +311759252 +1093402624    +9769363 -1499726735   -43325844
        +50  -412375444 +1362953104  +211095956 +1630011392    +9769364 +1766064128    +9746836 +1901461504    +9769364
+2034040832    +9769364   -23331727   -43297172  -157549455  -412422548  -290521088    +9738644  -425853840 +2123706772
+2147483647    +9767316  -959054737  -580194708  -824967168    +9767316  -960561152    +9769364 -1095892992    +9769364
-1229258752    +9769364 -1363214336    +9769364 -1497170833  +445976980 -1633944463   -43323796 -1768031121  +949293460

*****************    JOB NAME : SKALA          EO :           DATE : 18/11/78-322    TIME : 15.32.38    *****************
*                                                                                                                          *
*  END OF STEP  *    STEP NO : 004    ELAPSED TIME : 0000.18                                                               *
*                                                                                                                          *
****************************************************************************************************************************

*****************    JOB NAME : SKALA          EO :           DATE : 18/11/78-322    TIME : 15.32.38    *****************
*                                                                                                                          *
*  END OF JOB   *    JOB TIME : 0004.59    CONNECTED TIME : 0009.07     STEP NUMBER : 004                                 *
*                                                                                                                          *
****************************************************************************************************************************
```

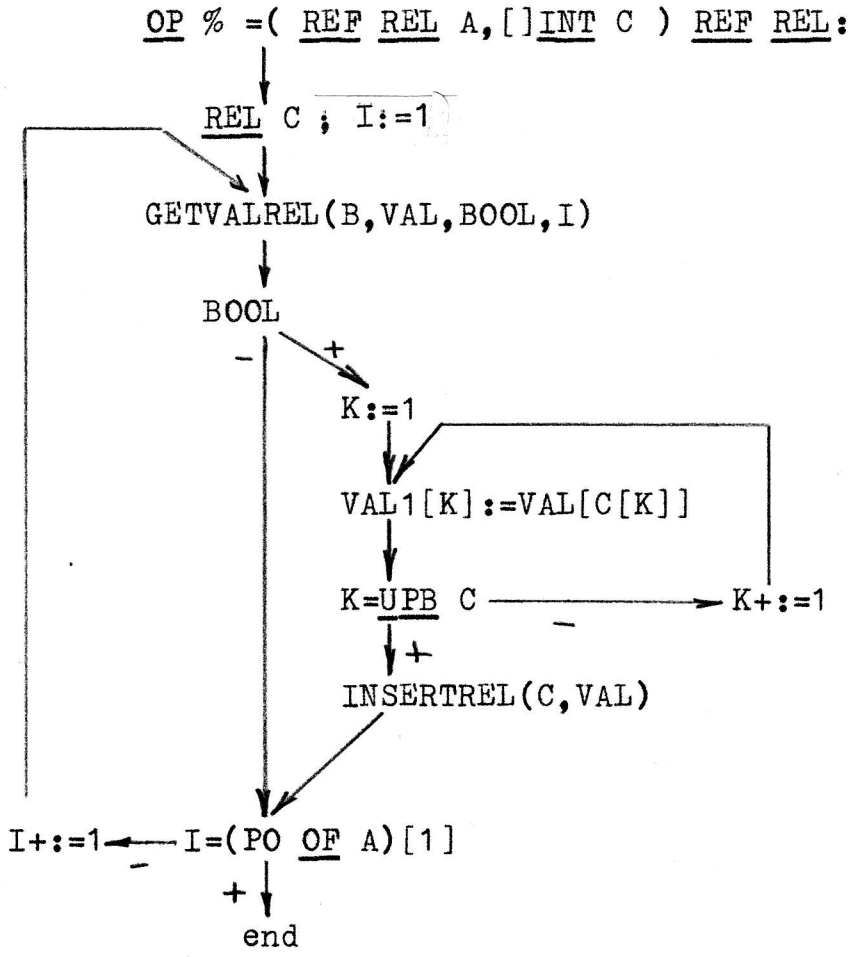
P Ř Í L O H A G

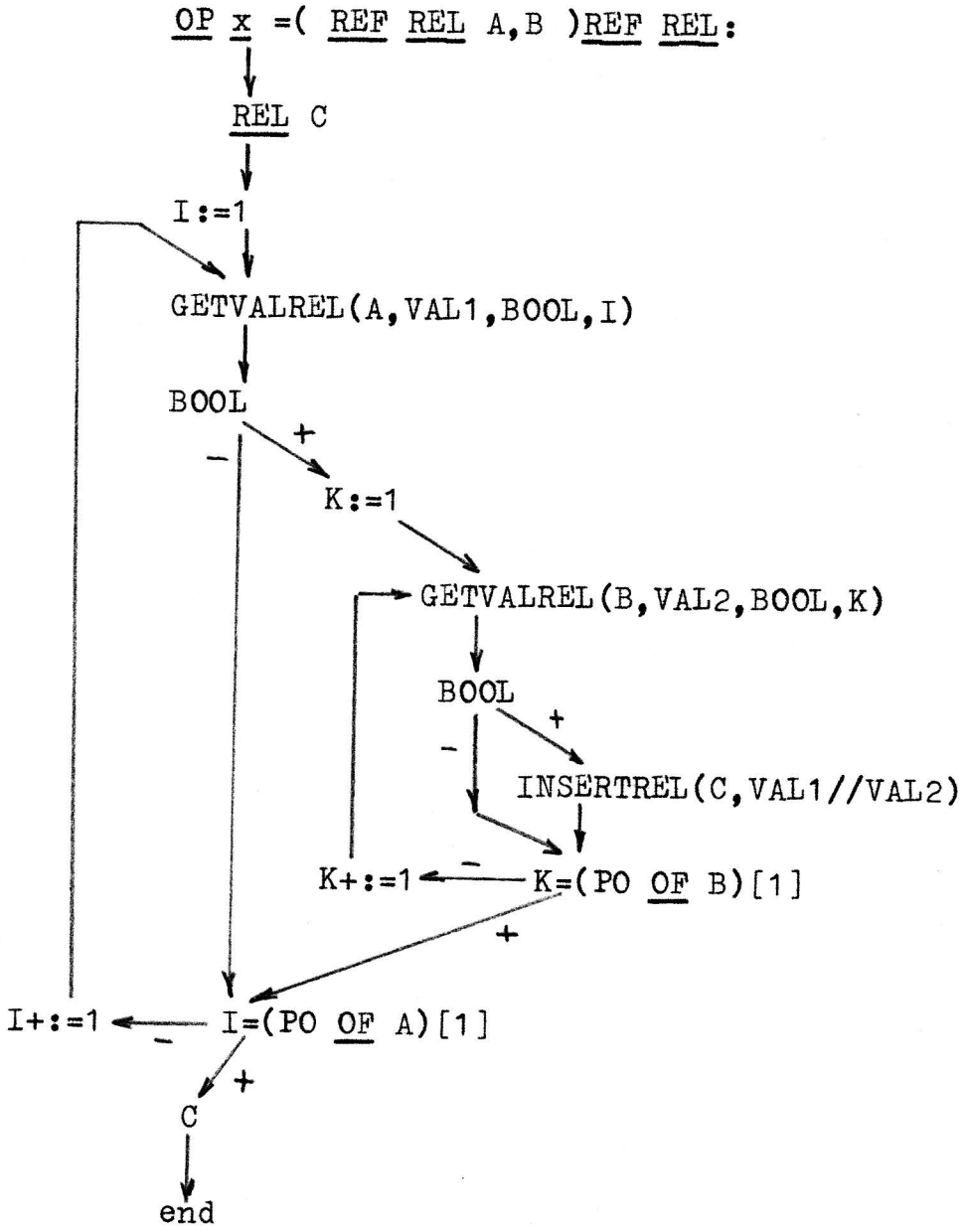
Některé operace nad relacemi


OP + =(REF REL A,B) REF REL:
A :=: B
+
-
A ∪ A = A
A
UPB DOM OF A =
UPB DOM OF B
-
PRINT("nekompatibilni relace")
+
REL C
C:=A
[1:UPB DOM OF B] STRING VAL
I:=1
GET VAL REL(B,VAL,BOOL,I)
BOOL
-
+
INSERT REL(C,VAL)
I+:=1
-
I = (PO OF B)[1]
+
C
end


OP - =(REF REL A,B) REF REL:
UPB DOM OF A =
UPB DOM OF B
PRINT("nekompatibilní relace")
REL C
A :=: B
A-A=∅
C:=A
[1:UPB DOM OF B] STRING VAL
I:=1
GET VAL REL(B,VAL,BOOL,I)
BOOL
DELETEREL(C,VAL)
I+:=1
I=(PO OF B)[1]
end

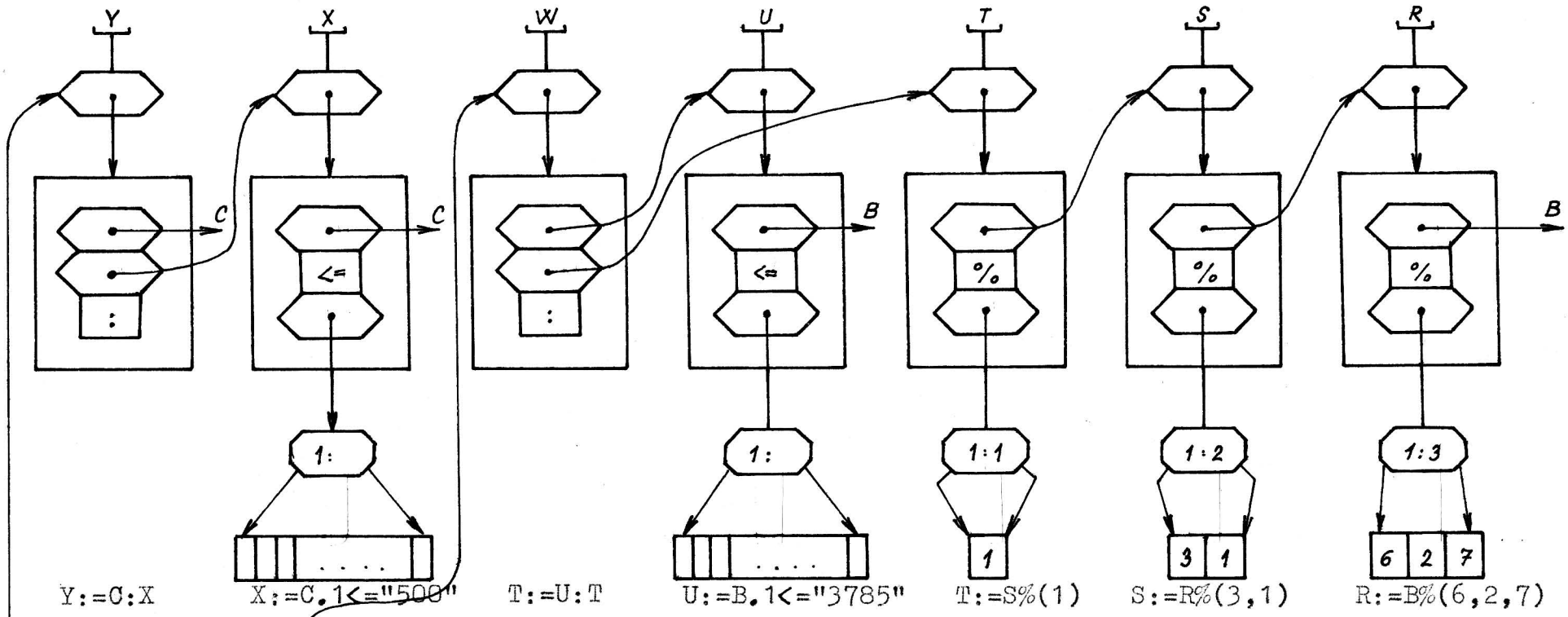

OP * =(REF REL A,B)REF REL:
A :=: A
+
A ∩ A = A
-
UPB DOM OF A =
UPB DOM OF B
PRINT("nekompatibilni relace")
-
+
REL C
I:=1
GETVALREL(A,VAL,BOOL,I)
BOOL
-
+
FIND REL(B,VAL,BOOL,L)
BOOL
-
+
INSERTREL(C,VAL)
I+:=1
I=(PO OF A)[1]
-
+
end

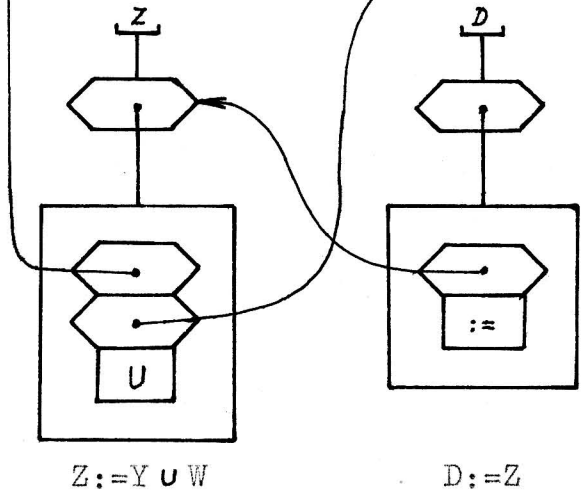

OP % =(REF REL A,[]INT C) REF REL:
REL C ; I:=1
GETVALREL(B,VAL,BOOL,I)
BOOL
-
+
K:=1
VAL1[K]:=VAL[C[K]]
K=UPB C
K+:=1
+
INSERTREL(C,VAL)
I+:=1
I=(PO OF A)[1]
-
+
end


OP x =(REF REL A,B)REF REL:
REL C
I:=1
GETVALREL(A,VAL1,BOOL,I)
BOOL
+
-
K:=1
GETVALREL(B,VAL2,BOOL,K)
BOOL
+
-
INSERTREL(C,VAL1//VAL2)
K+:=1
-
K=(PO OF B)[1]
+
I+:=1
-
I=(PO OF A)[1]
+
C
end

P Ř Í L O H A H

Realizace dotazu uživatele pomocí proměné módu cursor





D:=B%(6.2.7)%(3,1)%(1):B.1 <="3785" ∪ C:C.1 <="500"

kde: B,C jsou proměnné módu rel
R,S,T,U,W,X,Y,Z,D jsou proměnné módu cursor

Dotaz uživatele před optimalizací