Jan K O Z Á K

Výzkumný ústav ekonomiky paliv a energetiky Praha

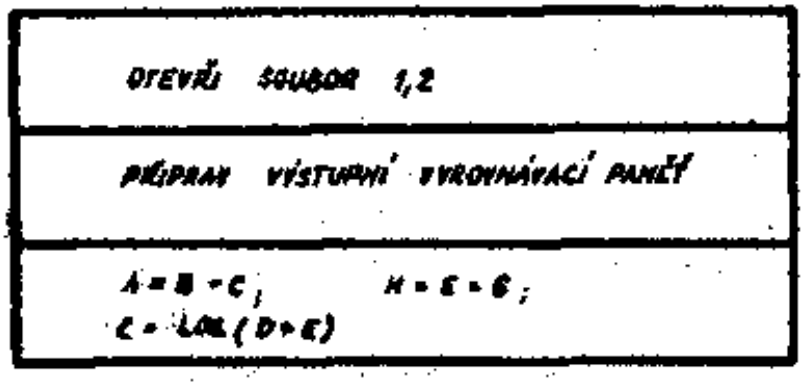
Grafické metody při tvorbě a dokumentaci programů

Grafické metody zůstávají jedním z nejoblíbenějších prostředků jak při tvorbě, tak i dokumentaci programů. Prohlubující se specialisace v oblasti software se projevila ve vývoji řady účelově zaměřených grafických zápisů.
Nejúživanějšími kategoriemi grafických metod jsou systémové diagramy, vývojové diagramy pro zápis algoritmů a strukturní diagramy pro popis dat.
Příspěvek se nezabývá klasifikací grafických metod, zaměřuje se na u nás méně známé grafické prostředky, které se osvědčily při vývoji moderních informačních systémů.

1. Vývojové diagramy pro strukturní programování

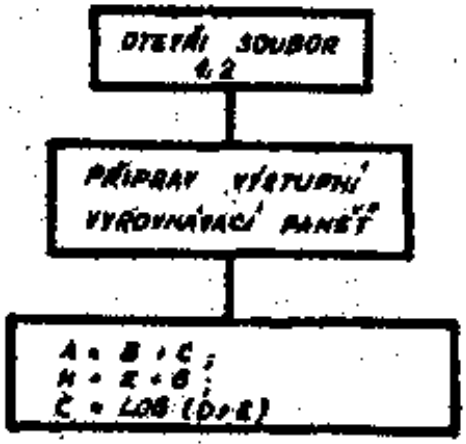
Hlavními charakteristikami strukturního programování je použití modulárního přístupu shora-dolů nebo zdola-nahoru a odstranění programovacího příkazu GO TO.
Současné standardisované vývojové diagramy jako je ČSN 36 9030 plně nevyhovují potřebám strukturního programování, neboť jejich přepis do programovacího jazyka je značně odlišný od původního grafického zápisu.
V poslední době se objevila celá řada variant vývojových diagramů, které nabízejí různé alternativy eliminace GO TO příkazů. Zde se soustředíme na diagramy popsané v /Chapin/, které autor podle sebe nazval Chapinovy, ačkoliv o jejich původu se v odborné literatuře vedla dlouhá polemika.
Základními prvky nového formátu vývojových diagramů jsou čtyři grafické značky /obdélník, spojnice, elipsa a kruh/ a tři symboly /lomítko, hvězdička a pravá šipka/.
Obdélník může mít jakoukoliv velikost, rozměr a orientaci, aby mohl pojmout dostatečný popis zobrazované funkce, jež

je částí zobrazovaného algoritmu. Posloupnost operací se předpokládá v diagramu od shora dolu, takže funkce jsou prováděny v posloupnosti jejich vertikálního uspořádání. Každý obdélník má jeden kontrolní vstupní bod a dále na konci jeden výstupní bod, jímž předává řízení další funkci.
Obdélník může zahrnout více operací, popř. sekvenční tok může být pro zdůrazdnění rozdělen do více sekcí /obdélníků/.
Obr. 1 a 2 zobrazují alternativy zobrazení sekvenčních operací



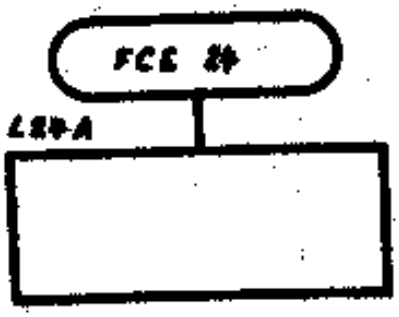
obr. 1 Obdélník tvořený ze tří základních obdélníků se sekvenčním průběhem operací

Uvnitř obdélníku symboly popisují jednotlivé operace. Konvenčně se předpokládá průběh operací od leva doprava a od shora dolu. Jednotlivé operace mohou být od sebe odděleny delimitačním znakem jako je čárka nebo středník.



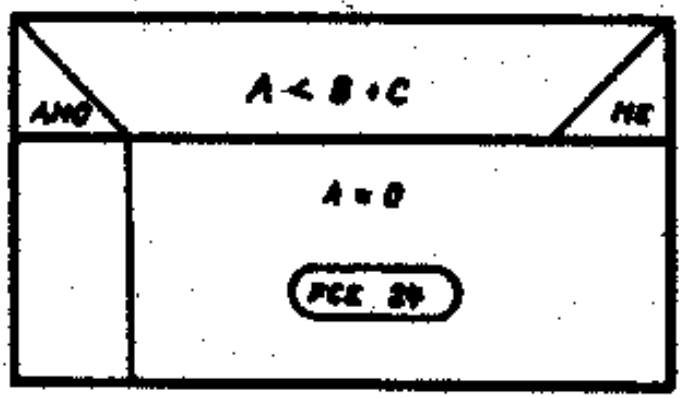
obr. 2 Jiná alternativa grafického zobrazení sekvenčního průběhu algoritmu. Části jsou explicitně pro zdůrazdnění rozděleny a spojení obdélníků je provedeno spojnicemi

Pojmenovaný vstupní bod je identifikován návěštím, které je zapsáno v levém horním rohu obdélníku. Toto jméno /mělo by být totožné jako ostatně další identifikátory v zápisu i v programu/ může být vynecháno, shoduje-li se se jménem funkce. Jméno funkce je zapsáno v elipse /obr.3/.



obr.3 Označení funkce elipsou a explicitní označení obdélníku návěštím

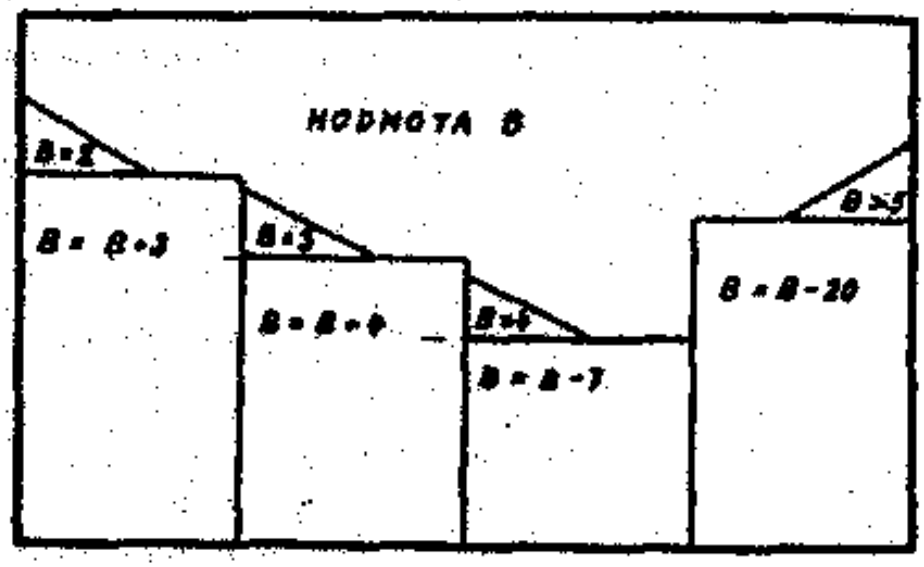
Výstupní bod z obdélníku může být explicitně vyznačen spojnicí vycházející ze spodní hrany obdélníku. Nevychází-li z obdélníku spojnice ani nenavazuje společnou hranou další obdélník, předpokládá se konec funkce, tj. z programátorského hlediska RETURN, návrat do funkce, která příslušnou funkci vyvolala.
Vícenásobný výstup z obdélníku jako je podmíněný přenos kontroly je proveden pomocí horizontálně uspořádaných obdélníků. Tyto obdélníky mají nad sebou obdélník s popisem podmínky a vyznačením výsledku testu.



obr.4 Podmíněný přenos typu if-then-else

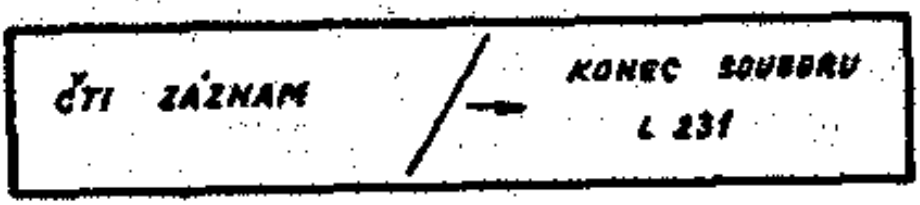
Na obr.4 je graficky zapsán PL/1 příkaz IF A<B+C THEN; ELSE DO; A=0; CALL FCE24; END;
Vícenásobný podmíněný příkaz, jehož programovacím ekvivalentem

je algolovský příkaz CASE, je tímto způsobem zobrazen na obr.5



obr.5 Vícenásobný podmíněný příkaz typu CASE

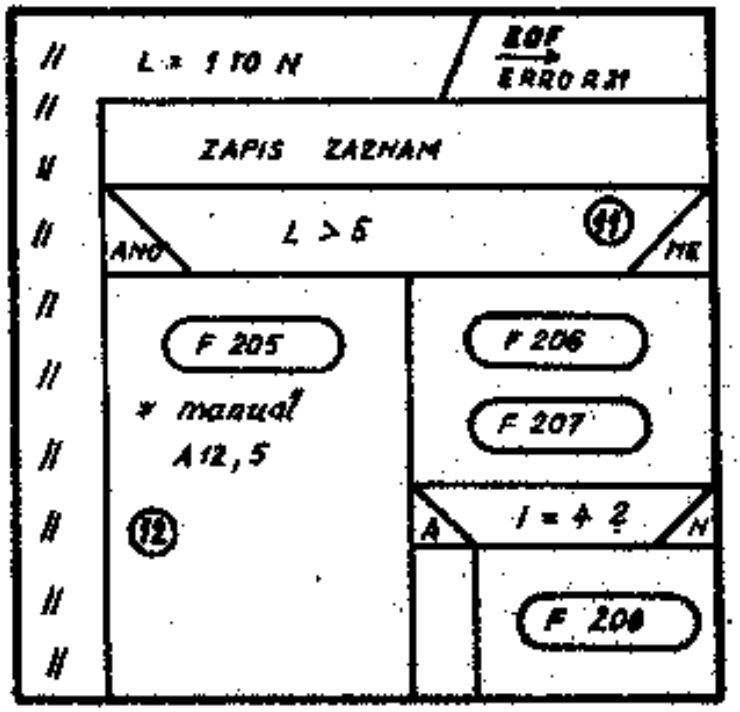
I když nepodmíněný skok bychom měli při návrhu algoritmu co nejvíce omezit, je nutné mít pro určité stavy /např. pro přerušovací systém/ jeho grafické zobrazení.
Nepodmíněný skok je zobrazen pravou šipkou následovanou jménem návěští, na které se předává programové řízení. Protože operace zahrnuje změnu řízení, je zobrazena zpravidla ve vlastním obdélníku.



obr.6 Nepodmíněný skok

Iterace, jejichž nejobecnější formou jsou iterace podmíněné testem /DO-WHILE/, jsou zobrazeny vnitřním obdélníkem označeným po levé straně symboly pro opakování- lomítky identifikující tak tělo smyčky. Nad vnitřním obdélníkem je prostor pro podmínky iterace. Konec vnitřního obdélníku označeného lomítky je konec iteračního bloku. Specifikovaná podmínka může mít kromě DO-WHILE tvaru i formu DO-UNTIL, resp. můžeme dalším nepodmíněným skokem indikovat rozdílný přenos řízení po

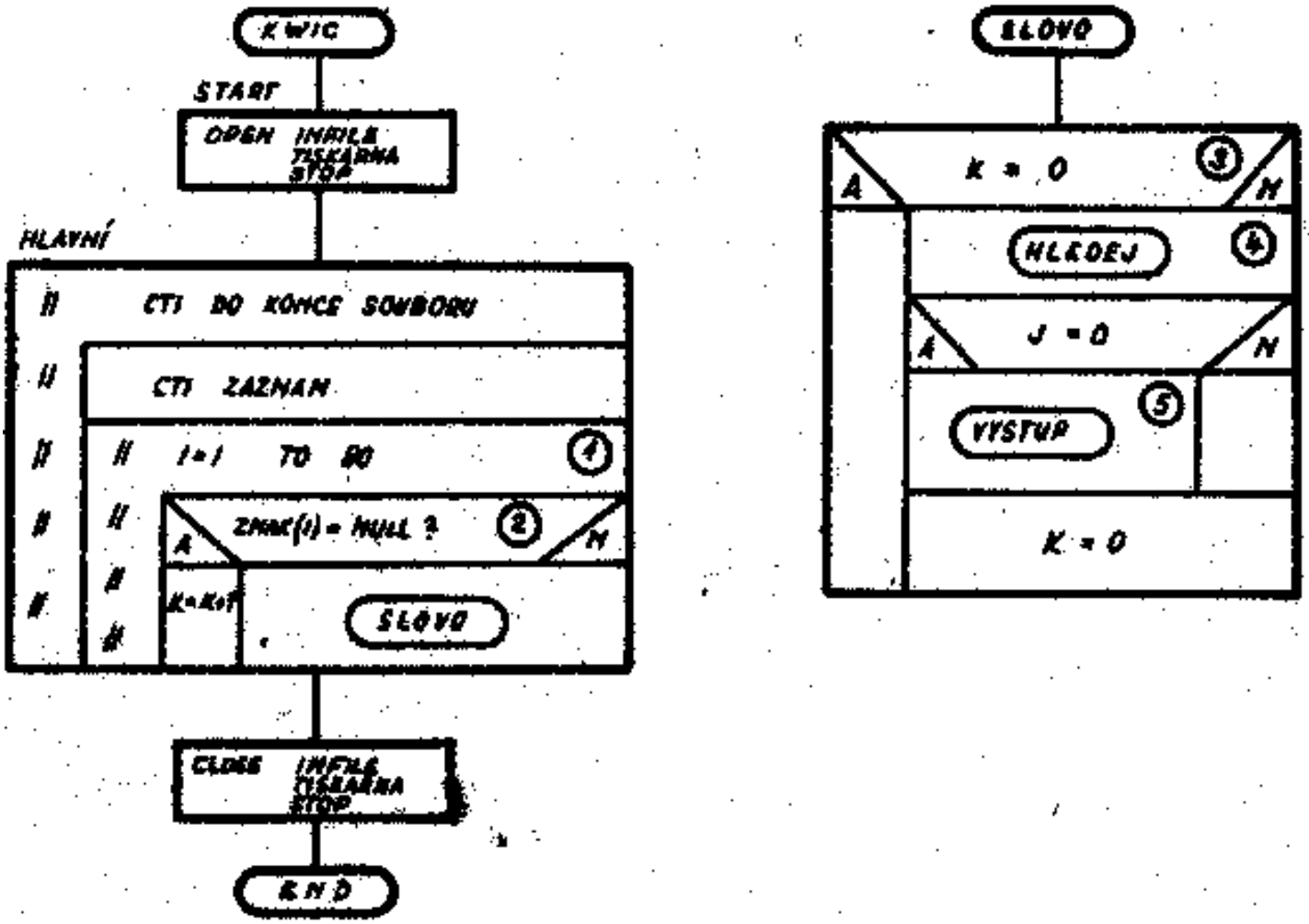
výstupu ze smyčky /normálně se předává řízení na další sousedící nebo spojnicí vyznačený obdélník/.



obr.7 DO-UNTIL smyčka s vyznačením další podmínky

Z obrázku 7 je zřejmé vyvolání funkcí pomocí explicitního uvedení elipsy se jménem funkce. Referenci obecných funkcí je lepší provádět, jak je obvyklé, odvolávkou pouze na jméno. Poznámky uvedené přímo ve vývojovém diagramu jsou označeny hvězdičkou. Ve výše uvedeném příkladu funkce F205 je uvedena v manuálu A12 na straně 5. Je-li místo přímo uvedené poznámky vysvětlivka dále v textu, je na ní odkaz s číslem uzavřeným do kroužku. Tak ve vysvětlivkách může být pod číslem 12 uvedeno manuál A12,5.

Jako příklad úlohy zapsané v tomto tvaru vývojového diagramu uveďme známý problém permutovaného rejstříku KWIC /Key Words In Context/.



obr.8 Příklad úlohy KWIC permutace textu

Vysvětlivky:

1/ Záznam obsahuje text označený ZNAK(1) ... ZNAK(80), u kterého se provádí permutace

2/ Delimitačním znakem NULL se rozumí mezera, tečka, čárka, středník, otazník atd.

3/ K je počítadlo počtu znaků právě permutovaného slova

4/ Funkce HLEDEJ provádí kontrolu na významnost slova. Nevýznamná slova jsou obsažena v souboru STOP. V případě, že hledané slovo tj. ZNAK(I-K) až ZNAK(I-1) není nalezeno v souboru, ukládá fce HLEDEJ do J hodnotu 0.

5/ Funkce VYSTUP edituje posunem tisk a provádí výstup na tiskárnu. Funkce HLEDEJ a VYSTUP nejsou zde uvedeny.

Z výše uvedeného příkladu je zřejmé, že nový formát vývojového diagramu svým zápisem přímo nutí uživatele k modulárnímu přístupu. Použije-li metody shora-dolu, sestrojí se nejdříve globální nástin algoritmu /programu/, jehož fce se postupně v dalších krocích rozepisují. Je důležité si připamatovat, že spojnice mezi obdélníky jsou v zásadě vertikální. Zápis je však přesto oproti standardizovanému formátu kompaktnější, použijeme-li zápis více funkcí vedle sebe. Je také přehlednější použitím menšího počtu konektorů. Z programátorského hlediska je jeho přepis do vyššího programovacího jazyka jednoznačný a snadný.
Nevýhodou tohoto formátu je, že nerozlišuje specifické operace, zařízení a použité média. Musí se proto doplnit systémovými diagramy např. v IBM verzi HIPO. Nelze také tento formát použít pro dokumentaci programů u těch organizací, které vyžadují dodržování čs. státních norem.

2. Strukturní diagramy pro popis dat

Zatím je obvyklé pracovat pouze s jedním typem záznamu v souboru, i když nemůžeme říci, že všechny systémy pracují pouze s jedním typem záznamu. Používají např. několik kmenových souborů nebo více transakčních či pracovních souborů.
U databázově pojatých systémů pracujeme pouze s jednou logickou databází, která je dále nějakým způsobem strukturně dělena. Struktura databáze je často složitá a proto si vypomáháme některými formálními prostředky pro její definici. Kromě matematického zápisu nebo informačních modelů - reference např. /Kozák/, je jedním z nejjednodušších a přehledných druhů popisu logické organizace databáze grafické zobrazení.
K tomuto účelu zavádíme pojem strukturního či Bachmanova diagramu /Lyon/.
Diagram specifikuje vztah mezi záznamy obdobným způsobem jako vývojový diagram činnosti a vazby procedur.
Zde použijeme značně modifikované Bachmanovy verze.
V naší verzi má strukturní diagram čtyři základní prvky: obdélník, spojnici se šipkou, hvězdičku a kruh.
Obdélník odpovídá typu záznamu. Jestliže databáze /resp. sou-

bor/ má n rozdílných typů záznamů, pak její diagram obsahuje n obdélníků.
Spojnice vyjadřuje vztah mezi záznamy. Protože záznamy v databázovém pojetí jsou vždy v hierarchickém vztahu, spojnice má šipku, která směřuje k podřízenému typu záznamu.
Hvězdička označuje přímo adresovatelné typy záznamů. Za hvězdičkou je případně uvedeno jméno přímo adresovatelného klíče. Jestliže adresace se provádí pomocí umístění záznamu, je tento způsob označen jednou hvězdičkou /např. index-sekvenční uložení, hash metoda uložení atd/.
Vytvoření extra indexů pro atribut záznamu je specifikováno dvěma hvězdičkami. V DBTG terminologii tomu odpovídají příkazy LOCATION MODE IS CALC, resp. SEARCH KEY IS ... /CODASYL/.
Kruh znamená výskyt záznamu. Tímto detailnějším způsobem specifikujeme určitá seskupení výskytů záznamů, která jsou přípustná nebo nedovolená.

obr.9 Strukturní diagram s jednou relací

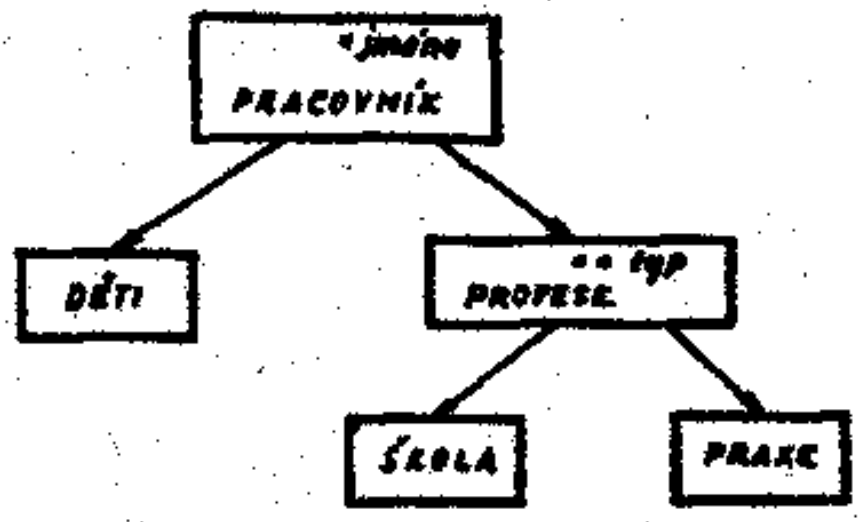
Na obr.9 je zobrazena pomocí strukturního diagramu velmi jednoduchá relace. Z tohoto diagramu můžeme vyčíst, že:

- existují dva typy záznamu PRACOVNÍK a DĚTI
- PRACOVNÍK a DĚTI jsou logicky svázány, přičemž PRACOVNÍK je vlastníkem /nadřazený/ relace a DĚTI je členem /podřízený/ relace
- PRACOVNÍK může vlastnit jakýkoliv počet záznamů typu DĚTI
- záznam DĚTI může mít pouze jeden záznam typu PRACOVNÍK jako vlastníka
- záznamy PRACOVNÍK jsou uspořádány v souboru způsobem, který umožňuje přímé adresování přes atribut "jméno". Adresovací metoda je pomocí lokace záznamu.

Pomocí strukturního diagramu můžeme specifikovat všechny druhy struktur. Na obr.10 je znázorněna kruhová struktura.

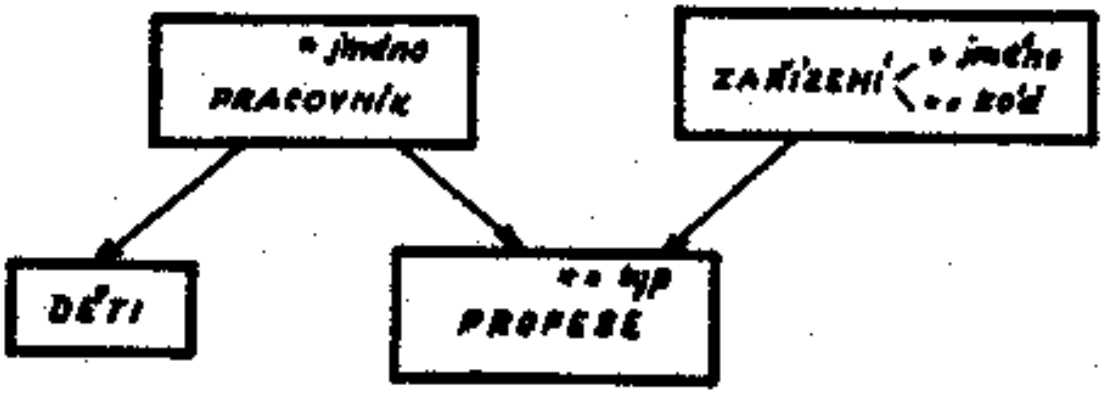
obr.10 Kruhová struktura

Kruhová /liniová/ struktura definuje rekursivní vztah. Rekurse vytvářející kruh může být specifikovaná přes více typů záznamů.
U stromové struktury prvek může mít jednoho nebo žádného vlastníka a naopak může vlastnit jakýkoliv počet dalších prvků.



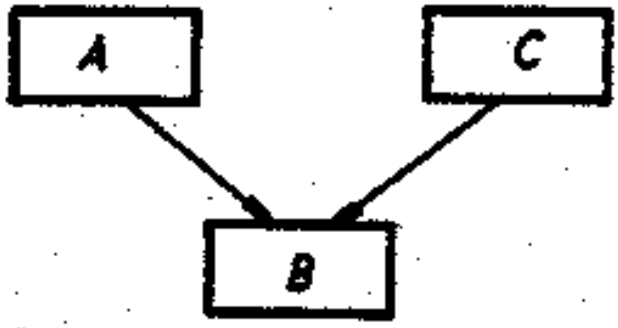
obr.11 Stromová struktura

U síťové struktury jeden prvek může mít jakýkoliv počet vlastníků a naopak může vlastnit jakýkoliv počet dalších prvků.



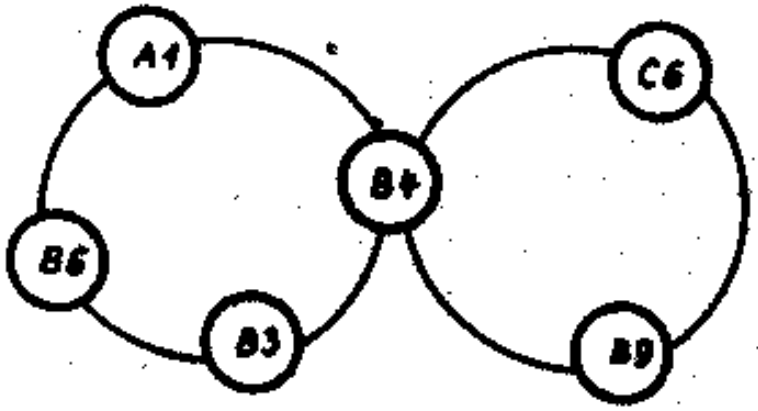
obr.12 Síťová struktura

Kruhem zobrazujeme některá speciální uspořádání. Např. běžná omezení většiny firemních systémů pro řízení bází dat /DBMS/ jako jsou IMS, DBTG atd. se projevují na úrovni výskytu u stromové /síťové/ struktury.
Tak pro strukturu zobrazenou na obr.13



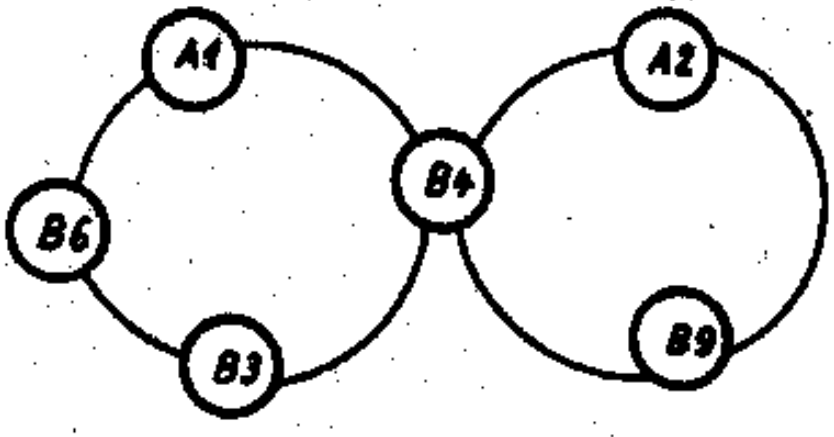
obr.13 Příklad stromové struktury

je přípustná struktura paměti podle obr.14, kdy jeden prvek lze zapojit do více výskytů každého z odlišných skupinových typů.



obr.14 Výskyt záznamů podle vztahu na obr.13

Není však přípustný výskyt záznamového typu zapojeného do dvou nebo více výskytů téhož skupinového typu, jak je ukázáno na obr.15.



obr.15 Jiný výskyt záznamů podle vztahu na obr.13

Strukturní diagramy jsou vhodné prostředky pro znázornění struktury databáze nebo složitějšího souboru. Doplňují tak ostatní prostředky jako je matematický popis nebo jazyk pro popis dat /DDL/ systémů pro řízení bází dat.

Závěr

Jsou nastíněny dvě grafické metody pro návrh software. Jejich předností je snadný grafický zápis, přehlednost a snadná aplikovatelnost. Jejich použití je výhodné při návrhu moderního programového zabezpečení.

Literature

Chapin N.: New format for flowcharts, Software & Experience, Vol 4, No.4 /October - December 1974/

Lyon J.: An introduction to data base design, Wiley - Interscience, New York 1971

Schubert R.: Directions in data base management technology, Datamation /September 1974/

CODASYL: DBLTG Journal of Development 1973, publikováno British Computer Society

Kosák J.: Metodika klasifikace a kódování informací, VÚEPE 1975