SPOJENÍ JACKSONOVY A GARDNEROVY TECHNOLOGIE STRUKTUROVANÉHO PROGRAMOVÁNÍ

Pavel Drbal VÚMS k.ú.o. ZAVT Praha

1. Úvod

Na minulých seminářích bylo uvedeno několik metod a technik, které pretendují na pokrytí celé metodiky programování, nebo alespoň její podstatné části. Jsou to techniky HIPO diagramy /7/, SPR /3/ a metody označované jmény autorů - Varnierova /4/ a Jacksonova /1/. Všechny čtyři jsou v naší republice užívány, okruh uživatelů je relativně úzký a žádná z nich zatím nezískala výraznou převahu. Mají významný společný rys - program vzniká hierarchickým rozkladem a je dobře strukturovaný /ve smyslu Dijkstry /5//.

Posuzujeme-li tyto metody z hlediska použití jako základu pro automatizační prostředky tvorby programů, upoutá naší pozornost tandem Jackson - Varnier. SPR a HIPO jsou pouze techniky, říkají nám, jak máme zapisovat hierarchický rozklad, kdežto metody Jacksona a Varniera navíc určují, proč máme program rozkládat tak a ne jinak - využívají myšlenky odvození programu z dat. Technika SPR se staví poněkud stranou také tím, že je ve svém základě trojrozměrná; metody, které se přizpůsobují dvourozměrnosti papíru, jsou ve výhodě. Metoda HIPO je velmi přizpůsobena lidskému způsobu myšlení, to způsobuje potíže při návrhu programové podpory, nakreslit HIPO diagram šablonkou je jednodušší, než jej zadat počítači k tisku. Existující programové podpora techniky HIPO /kreslení HIPO diagramů - existuje v NDR/ nevyvolala u uživatelů této techniky žádné nadšení.

Výše uvedené úvahy nelze v žádném případě chápat jako útok proti technikám HIPO a SPR, pouze se zde ukazuje, že jsou zaměřeny spíše na výlučné používání člověkem, jejich automatizace je pravděpodobněji obtížnější, než je třeba. Další metody zde neuvádím, protože jsou buď specializované, nebo jejich automatizační prostředky již existují /normalizované programování - RPG, rozhodovací tabulky - PROTAB/.

Při srovnávání metod nebudeme mluvit o Varnierově metodě, ale o Gardnerově verzi této metody /2/. Gardner ve své knize říká, že ideje jsou Varnierovy, on že jenom tuto metodu přizpůsobil anglosaskému způsobu myšlení. Diskutovat o tom, zdali je jedna metoda lepší než druhá, není zrovna nejužitečněji strávený čas, považuji za vhodnější spojit užitečné rysy obou metod v jeden celek. Tento článek je, pokud je mi známo, prvým takovým pokusem.

V jednom článku nelze popsat i srovnat obě metody, popis metod je ve dvou knihách s celkem 500 stránkami textu. Budu spokojen, jestliže tento text poslouží čtenáři ovládajícímu jednu metodu k orientaci v metodě druhé. Nedostatkem tohoto článku je malé množství příkladů. To lze řešit jen tak, že vystoupení na semináři bude z větší části věnováno příkladům srovnání jednotlivých bodů.

Obecně lze obě metody charakterizovat takto:
Jacksonova metoda je souhrn různých navzájem relativně nezávislých obratů a technik spojených v jeden systém; kdežto Gardnerova metoda je kompaktní, logicky sevřený celek, snažící se nahradit různé techniky jednou zobecněnou. Základní východiska jsou stejná: strukturované programování, tři základní typy hierarchického rozkladu /sekvence, selekce a iterace/, vyjádření struktury dat i programu stromovými grafy, odvození struktury programu ze struktury dat.

Nelze zde uvádět všechny rozdíly, při daném rozsahu článku by to vedlo k povrchnosti, budou zde uvedeny jen vybrané výrazné rozdíly. Hodnotit budeme spíše ve směru, jak Gardnerem ovlivnit Jacksona. Důvodem pro tento přístup je to, že již existuje automatizační prostředek pro technologii vycházející z metodiky Jacksona /systém DOGA - Dokumentační a generační prostředek pro automatizaci strukturovaného programování, byl vyvinut ve VÚMS, viz /6//. Závěrem bude metodické doplnění Jacksonovy metody i návrhy na doplnění či změnu systému DOGA.

2. Porovnání postupů

Nejprve porovnáme postupy obou metod, v dalším textu budeme některé rozdíly probírat podrobněji. Body postupu budeme označovat písmeny J nebo G podle příslušnosti k metodě, v závorce uvedeme body postupu druhé metody, které plní tutéž nebo podobnou funkci. J- nebo G- označuje, že druhá metoda nemá odpovídající bod postupu.

Gardner:

G1. Sestavení výstupní datové struktury. /J1/

G2. Sestavení vstupní datové struktury. /J1/

G3. Kontrola, zdali vstupní datová struktura odpovídá výstupní datové struktuře. /J1/

G4. Sestavení doplňkových /odvozených/ datových struktur. /J1/

G5. Řešení složitých selekcí. /J-/

G6. Sestavení programové struktury. /J2/

G7. Kontrola programové struktury, zdali odpovídá struktuře výstupních dat. /J1,J2/

G8. Doplnění struktury převodu fyzického vstupního záznamu na logický záznam. /J1,J2, resp. J5/

G9. Tvorba seznamu instrukcí dle jednotlivých typů a určení, kam která instrukce bude umístěna. /J3,J4/

G10. Zjištění "degerovaných" komponent programové struktury. /J-/

G11. Umístění instrukcí do struktury ve správném pořadí. /J4/

G12. Kontrola umístění instrukcí zajišťujících výstup proti struktuře výstupních dat. /J-/

G13. Kódování. /J6/

Jackson:

J1. Sestavení datových struktur /vstupních i výstupních/ a zjištění, zdali si odpovídají. V případě, že si neodpovídají, úprava zadání. /G1,G2,G3,G4,G8/

J2. Sestavení programové struktury. /G6,G7,G11/

J3. Tvorba seznamu podmínek a operací. /G9/

J4. Umístění operací do struktury. /G9,G11/

J5. Přepis do pseudokódu. /G-/

J6. Kódování. /G13/

K Jacksonově metodě patří navíc předpisy, jak se vypořádat s případy, kdy dojde k rozporu struktur /tj. vstupní a výstupní struktury si neodpovídají/, tyto předpisy se liší podle typu rozporu. Podrobnost metod je přibližně stejná, i když Gardnerův předpis postupu je vyjádřen více body.

Podobnost obou metod je střejmá. Gardnerova metoda obsahuje navíc bod G5 - "Řešení složitých selekcí", který lze bez jakéhokoliv kontextu převzít do libovolné metody. V podstatě se jedná o to, že se u složité podmínky určí pomocí Veitchových diagramů v jakém pořadí se mají jednotlivé jednoduché podmínky testovat, aby program byl optimální. Jaksonova metoda obsahuje navíc bod J5 - "Přepis do pseudokódu". Gardner totiž prosazuje soustavné používání podprogramů, kdežto Jackson připouští oba tvary zápisu v programovacím jazyce, jak s použitím podprogramů, tak i bez nich /tzv. perform-free forma/. Bod J5 slouží k usnadnění přepisu struktury programu do lineární formy zápisu v programovacím jazyce.

1. Struktury

Obě metody znázorňují struktury dat i programů jako stromové grafy, jeden ze základních rozdílů je v chápání komponent těchto struktur. U Jacksona jsou všechny komponenty jednoho druhu - logicky rovnocenné - a čtyř typů /sekvence, selekce, iterace a koncová komponenta - list grafu - která obsahuje buď proměnné /u datových struktur/ nebo výkonné operace /u programových struktur//.
U Gardnera je situace poněkud složitější, zjednodušeně lze popsat takto: Datové komponenty jsou dvou druhů, jednoho, který ovlivňuje strukturu programu - takové komponenta je nazývána množina; druhého, který neovlivňuje strukturu programu. Struktura programu je odvozována pouze ze struktury množin, a instrukce nejsou do struktury programu umisťovány. Můžeme si to představit tak, že základní jednoprvkovou množinou je záznam /například řádek na tiskárně/. Další množiny jsou tvořeny seskupováním této základní množiny, a to buď jako iterovaná nebo alternativní množina. Struktura řádku - popis proměnných - může být také vyjádřena v datové struktuře, ne však jako množiny, ale jako komponenty druhého druhu. Z komponent prvého druhu - množin - se odvozuje struktura programu, z komponent druhého druhu se odvozují instrukce /např. naplňující proměnné tištěného řádku/. V zápisu datových struktur orámovaný uzel označuje množinu, druhý druh komponent není orámován. Srovnávací příklad viz na obrázku 1. Struktura programu má u Gardnera pouze orámované uzly, a instrukce lze přidělovat ke každému uzlu. U Jacksona mohou

být komponenty jen čtyř výše uvedených typů, výkonné operace lze přidělovat jen na koncové uzly /listy stromového grafu/. Z toho vyplývá nutnost vkládat pomocné uzly do grafu struktury programu, viz srovnávající příklad na obrázku 2. Gardnerův způsob zápisu umožňuje, aby jedna množina měla několik podmnožin, z nichž některé mohou být iterativní, jiné alternativní, viz obrázek 3.

4. Instrukce a operace

Rozdíly mezi pojetím struktur se vážou s rozdílem mezi Gardnerovými instrukcemi a Jacksonovými organizačními a výkonnými operacemi. Gardnerovy instukce odpovídají příkazům vyššího programovacího jazyka /např. Cobolu/, Jacksonovy operace jsou obecnější povahy. Jackson dělí operace na dva typy, organizační a výkonné, kde organizační operace /operace zajišťující předání řízení/ jsou jednoznačně dány strukturou programu, takže programátor určuje jen podmínky /ukončení iterace, výběr větve selekce/, výkonné operace jsou všechny ostatní, mezi jinými též operace zajišťující testovatelnost podmínek. Z hlediska programovacího jazyka lze říci, že organizační operace jsou příkazy cyklu, podmíněný příkaz, skokový příkaz a deklarace návěští; výkonné operace jsou přiřazovací příkaz, příkazy pro soubory, vyvolání podprogramu.

Gardnerovy instrukce jsou rozděleny podrobněji do šesti typů, jsou to instrukce vstupu dat, předání řízení /včetně zadání cyklů/, příprava pro předání řízení, výpočty a jejich příprava, výstup dat a jejich příprava, příprava a zajištění rekurse. Jackson celkem neurčitě říká: "seznam výkonných operací vytvoříme dle požadavků úlohy", kdežto Gardnerovo dělení s podrobnějšími doporučeními je daleko lepším vodítkem pro tvorbu seznamu. Osobně považuji tento rozdíl za nejdůležitější a doplnění Jacksonovy metody o podrobnější předpisy pro vytváření seznamu operací za velmi žádoucí.

5. Přiřazení příkazů struktuře

V Jacksonově metodě existuje graf struktury programu ve dvou tvarech - bez operací a s operacemi. Přiřazení operací do struktury se chápe jako další hierarchický rozklad listů struktury na sekvenci výkonných operací; všechny organizační operace jsou úplně vyjádřeny strukturou a přiřazením podmínek iterovaným a vybíraným komponentám. Operace /či podmínka/ se v seznamu vyskytuje jen jednou a může být přiřazena na několik různých míst struktury. Seznam operací

a podmínek spolu s programovou strukturou a přiřazenými čísly operací a podmínek je úplný /funkční/ popis programu.

V Gardnerově metodě je jiný postup. Všechny uzly programové strultury se očíslují. V seznamu instrukcí se u instrukce uvede pořadové číslo uzlu programové struktury, to znamená, že instrukce se v seznamu uvede tolikrát, ke kolika uzlům patří. Struktura programu spolu se seznamem instrukcí netvoří popis programu, není totiž určeno pořadí instrukcí příslušných jednomu uzlu. Toto pořadí se určí až při kódování. Pořadí instrukcí odpovídajících výkonným operacím je dáno předpisem /G11/, pro umístění instrukcí předání řízení není předpis, Gardner neurčitě říká: "jejich umístění nečiní v praxi potíže".

6. Kódování

Porovnáme-li převod z tvaru "struktura programu a seznam operací či instrukcí" do tvaru "zápis v programovacím jazyce", vynikne zásadní rozdíl mezi oběmi metodami. U Gardnera poprvé program vznikne až v zápisu v programovacím jazyce, struktury a seznam instrukcí je pomůcka pro vznik tohoto zápisu. U Jacksona vznikne program již v kroku J4 v zobecněném zápise /s neformálním popisem funkce operací/ a dále se program již jen postupně převádí do zápisu v programovacím jazyce - a to ve dvou krocích. Prvý krok /J5 - - Přepis do pseudokódu/ převádí strukturu programu do lineárního zápisu a vkládá popis výkonných operací, v druhém kroku /J6 - - Kódování/ se provádí převod neformálního popisu operací do formalizovaného zápisu programovacího jazyka. Jestliže popis operací formalizujeme již v bodě J4, je provádění kroků J5 a J6 čistě mechanické, lze jej tedy automatizovat. Oproti tomu provádění bodů G9 až G11 vyžaduje účast člověka.

Přínosem Gardnerovy metody je bod G11, kde rozděluje instrukce do dvanácti skupin a formuluje pravidlo, jak musí být seřazeny instrukce různých skupin patřící k téže komponentě.

7. Datové struktury

Obě metody vycházejí z téže ideje, nejprve se popíší datové struktury a pak se z nich odvozuje programová struktura. Jeden ze zásadních rozdílů mezi metodami je způsob tohoto odvození. Proces odvození programové struktury z datových struktur lze rozdělit do dvou částí. Je to základní odvození, tj. když nebyl zjištěn rozpor

mezi datovými strukturami, a řešení rozporu datových struktur. Základní kriterium je u obou metod stejné, odpovídající komponenty datových struktur se musí shodovat počtem svých reprezentantů i jejich pořadím.

Jacksonův základní postup je tento: Vytvoříme popisy datových struktur. Data popisovaná jednou strukturou odpovídají přibližně souboru, to znamená, že k jednomu programu může existovat několik struktur vstupních nebo výstupních dat. Pokud nejsou struktury v rozporu, vytvoří se z nich programová struktura jakýmsi sjednocením. Pro toto "sjednocení" existuje sedm pravidel. Pokud nelze aplikovat ani jedno z těchto pravidel, dochází k rozporu struktur. Dá se také říci, že programová struktura obsahuje podstruktury, které korespondují datovým strukturám.

V Gardnerově základním postupu se nejdříve vytvoří popis struktury výstupních dat, pak popis struktury vstupních dat. Programu přísluší jen dvě datové struktury, jedna vstupní a jedna výstupní. Vstupní datová struktura je prohlášena za programovou strukturu. Gardner říká: Vstupní datová struktura určuje programovou strukturu, výstupní struktura určuje instrukce. Jakýkoliv nesoulad mezi vstupní a výstupní strukturou je řešen jako rozpor struktur.

Jackson dělí rozpory struktur do několika typů /v knize /1/ jsou uvedeny tři/, každý řeší zvláštní technikou, která je obecně známa /třídění, inverze programu/ a použitelná i samostatně. Předpisy pro řešení rozporů jsou podrobným návodem.

Gardner řeší všechny rozpory datových struktur zavedením doplňkových /odvozených, derivovaných/ dat. Technika doplňkových dat je obecná, řeší širokou třídu příkladů. Její znalost je důležitá pro pochopení složitějších problémů. Zavedení doplňkových dat osvětlím na následujícím velmi jednoduchém příkladě.

Vstupní soubor jsou štítky, které jsou setříděny dle čísla dílny. Výstupem je opis těchto štítků /jeden štítek - jeden řádek/, navíc je za řádky jedné dílny uveden součtový řádek /např. součet odpracované mzdy/. Výstupní struktura je tedy iterace dílen, každá dílna se skládá jednak z iterace opisu štítků této dílny, jednak ze součtového řádku. Vstupní struktura je iterace dílen, každá dílna je iterace štítků této dílny. Potud je postup obou metod stejný. Jacksonova metoda vytvoří strukturu programu jako sjednocení obou struktur, tj. struktura programu bude odpovídat

výstupní datové struktuře, neboť vstupní struktura odpovídá podstruktuře výstupní struktury. V Gardnerově metodě nemůžeme prohlásit vstupní strukturu za programovou, protože v ní není komponenta odpovídající součtovému řádku. Musíme tedy vytvořit doplňková data umožňující řešit tento rozpor. Tato doplňková data budou představována proměnnou SUMA v operační paměti. K těmto datům vytvoříme vstupní doplňkovou strukturu a výstupní doplňkovou strukturu. Výstupní doplňková struktura bude popisovat naplnění proměnné SUMA, vstupní doplňková struktura bude popisovat používání proměnné SUMA.

Poznámka. Uvědomte si rozdíl oproti vstupní a výstupní datové struktuře. Program v zásadě nejprve čte vstupní data a na jejich základě vytváří výstupní data, ale doplňková data jsou nejprve vytvářena a teprve po vytvoření čtena.

V našem příkladě jsou vstupní i výstupní doplňková struktura vlastně stejné, ve složitějších případech tomu tak není. U vstupní doplňkové struktury určíme frekvenci použití - jedenkrát za dílnu - - a připojíme ji do vstupní datové struktury na místo s touž frekvencí. Tím jsme dostali vstupní datovou strukturu, která již plně koresponduje s výstupní datovou strukturou, a můžeme ji tedy prohlásit za strukturu programu. Z výstupní doplňkové struktury odvodíme instrukce nulování proměnné SUMA a přičítání sčítance.

Výsledkem obou metod je stejná programová struktura. Jacksonova metoda má jednodušší postup, Gardnerova nám navíc dá představu, jaké instrukce /operace/ použijeme. Gardnerova metoda je kompaktnější, logičtější, nedává nám však návod, pro konkrétní případ musíme postup odvozovat zvlášť. Jacksonova metoda neformuluje obecnou teorii, uvádí pouze jednotlivé typy rozporu struktur a návod, jak tento rozpor vyřešit. Typ rozporu se určí porovnáním datových struktur. Jacksonem uvedené rozpory struktur pokrývají velmi velkou oblast programování, těžce se hledá příklad, který je metodou nepokrytý.

Zřejmý rozdíl je vidět na příkladech v obou knihách, /1/ a /2/, které odpovídají "rozporu mezí", druhému typu rozporu dle Jacksonovy klasifikace. Jackson z datových struktur odvodí jak typ rozporu, tak strukturu vloženého souboru, rozdělí úlohu na dvě podúlohy, řeší každou samostatně, a na závěr jeden program invertuje v podprogram druhého programu, tj. obě nezávisle vzniklá řešení spojí mechanicky v jedno řešení. Gardner při obdobné úloze prohlásí a priori to, čemu Jackson říká vložený soubor za logický vstup

/resp. logický výstup/ a řeší jednak vlastní program, jednak podprogram, který převádí fyzický vstup na logický vstup /resp. logický výstup na fyzický výstup/. Gardner neuvádí kdy a proč se použije tento postup. Přes zdánlivou podobnost vloženého souboru Jacksona a doplňkových datových struktur Gardnera není řešení rozporu mezi pomocí doplňkových struktur nasnadě.

Na druhé straně technika doplňkových struktur usnadní řešení úloh, které nespadají úplně přesně do Jacksonovy klasifikace. Jsou to případy /podle dosavadních zkušeností/, kdy se rozpor v pojetí Jacksona netýká dat jako celku, ale pouze malých částí, takže je zvládnutelné např. nahrát postupně tyto části dat do operační paměti a tam je třeba přetřídit. Při dogmatickém chápání Jacksona bychom byli nuceni přetřídit celý soubor, což je samozřejmě náročnější na výpočetní kapacitu.

8. Shrnutí a závěry

Obě metody lze považovat za ekvivalentní a volba mezi nimi je více věcí osobního vkusu než věcných rozdílů. Zásadní rozdíl vidím ve vhodnosti pro automatizaci. Jacksonova metoda umožňuje popsat celý program na úrovni struktury programu, Gardnerova až na úrovni programovacího jazyka.

Na základě Jacksonovy metody existuje několik automatizačních prostředků, program se zadává buď na úrovni pseudokódu /automatizace bodu J6/, jsou to předkompilátory nebo systémy maker určitého jazyka, nebo na úrovni struktury programu /výše zmíněný systém DOGA, automatizace bodů J5 a J6/.

Není mi znám žádný systém, vycházející z Gardnerovy nebo Varnierovy metody, lze si však představit interaktivní systém, který setříděním instrukcí podle čísla komponenty a typu instrukce vytvoří základ programu. Tento program pak programátor interaktivně doplní o instrukce předání řízení /jejichž pořadí nelze zatříděním jednoznačně určit/.

Doplnění Jacksonovy metody idejemi Varniera a Gardnera:

1. Rozdělit seznam výkonných operací na následující části:
   - Seznam vstupních operací.
   - Seznam výstupních a výstup připravujících operací.
     Pravidlo: Každému listu výstupní struktury musí být přiřazena alespoň jedna operace tohoto podseznamu.

- Seznam operací pro přípravu podmínek. Pravidlo: Každá proměnná použitá v podmínkách musí být naplněna buď operací čtení nebo operací tohoto podseznamu.
- Seznam výpočetních a pomocných operací. V tomto podseznamu jsou operace zajišťující vlastní výpočty úlohy.

2. Při tvorbě datových struktur zapisovat nejdříve výstupní datové struktury, pak vstupní. Tento postup umožní lépe pochopit požadavky úlohy a přizpůsobit jim popis vstupní datové struktury.

3. Při umisťování operací do struktury se řídit frekvencí operací /operaci, která se má provádět n-krát, umístit do podstruktury se stejnou frkvencí provádění/ a při umisťování více operací k jednomu listu struktury se řídit pravidly G11.

4. Doplnit techniku doplňkových struktur.

5. Případně lze také uvažovat o použití Veitchových diagramů v případě složitých podmínek, tj. rozlišit selekci /v programové struktuře bez přiřazených operací/ od realizace selekce /v programové struktuře s přiřazenými operacemi/.

Návrhy rozšíření automatizačních prostředků vycházejících z Jacksonovy metody:

1. Zobrazit strukturu programu ve dvou verzích:
   - podrobné, odpovídající Jacksonově struktuře s operacemi,
   - přehledné, odpvídající Gardnerově struktuře množin.
   Tento postup je zvláště vhodný u složitých programů.

2. Kontrolovat pořadí přidělených operací podle typů operací /analogie bodu G11/. Vhodnost tohoto návrhu není úplně zřejmá, je nebezpečí, že pracnost určování či zadávání typu operace převýší užitečný efekt.

3. Dát možnost odlišit operace vzniklé z doplňkových datových struktur od ostatních operací, např. odlišit operace pracující s přepínači od vlastních výkonných operací.

4. Uvážit možnost tvorby automatizačních prostředků pro zpracování Veitchových diagramů, nebo možnost zařazení aparátu rozhodovacích tabulek do automatizačních prostředků.

\- - -

Pokud čtenář došel k závěru, že žádná metoda nemůže spasit programování, ale že již existuje několik způsobů, jak jej zefektivnit, potom článek splnil svůj účel.

9. Literatura

/1/ Jackson, M.A.: Principles of Program Design, Academic Press, London 1975

/2/ Gardner, A.C.: Practical LCP, McGraw-Hill, London 1981

/3/ Nichtburger,E.: Strukturované programování metodou SPR, seminář PROGRAMOVÁNÍ '80, Ostrava 1980

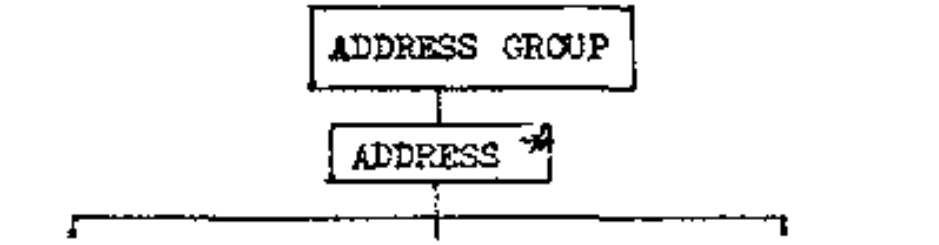
/4/ Veselý,P.: Metoda "Logical construction of programs", seminář PROGRAMOVÁNÍ'80, Ostrava 1980

/5/ Dahl,O.J.,Dijkstra,E.W.,Hoare,C.A.R.: Structured programming, Academic Press, London 1972

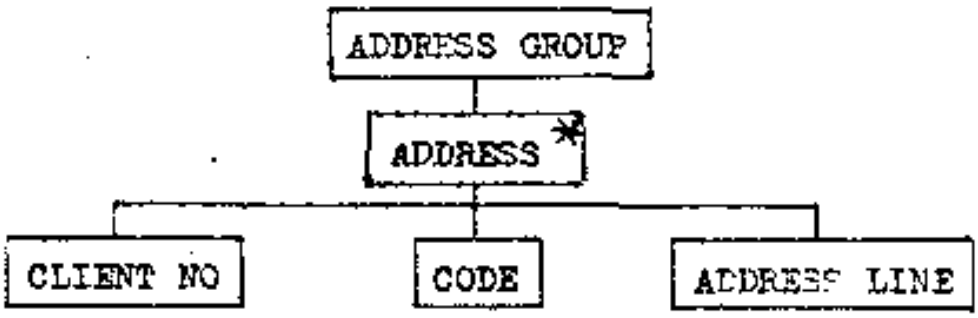
/6/ Drbal,P.: Programová podpora strukturovaného programování, seminář PROGRAMOVÁNÍ '82, Ostrava 1982

/7/ Nepovím,M.: Strukturované projektování, metodické materiály ČSVTS při Stavebních závodech Praha, Praha 1981

Gardner:

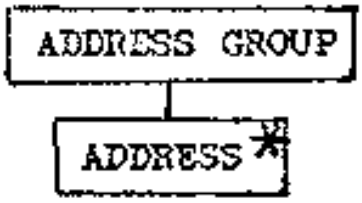


Jackson:

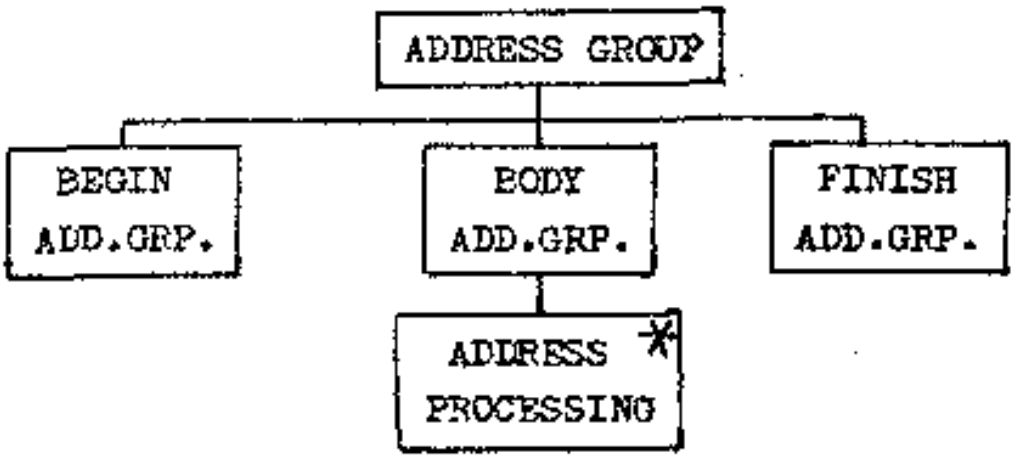


Obr. 1. Znázornění datových struktur.

Gerdner:



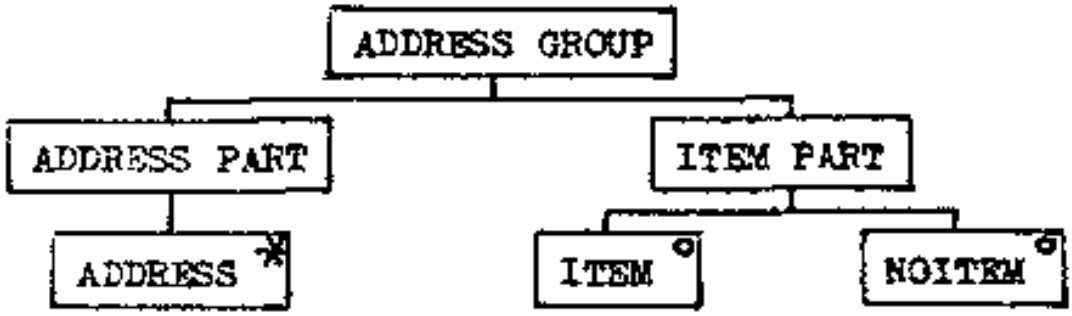
Jackson:



Obr. 2. Znázornění programových struktur.

Gerdner:

Jackson:



Obr. 3. Znázornění programové komponenty s různými typy podkomponent.