Edvard NICHTBURGER
VC ČKD Praha

STRUKTUROVANÉ PROGRAMOVÁNÍ METODOU SPR

Úvod

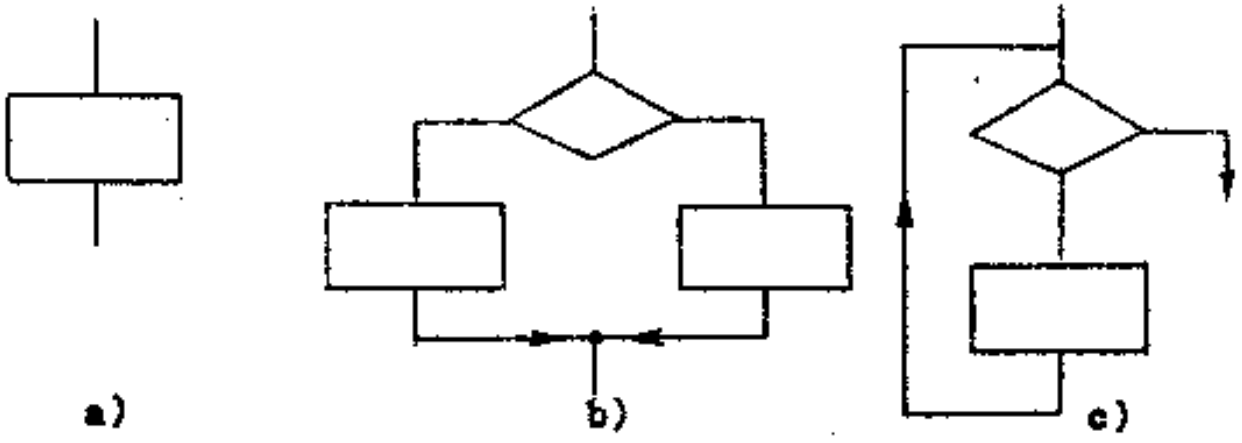
Již řadu let se hovoří o strukturovaném programování, programování shora dolů, postupném zjemňování, apod. V tomto referátu uvedeme stručný popis těchto metod, seznámíme se s metodou diagramů SPR.

Zkratka SPR vznikla ze slov Sequence, Parallelism, Refinement, česky bychom mohli říkat třeba Sled příkazů, Postupné zjemňování, Rozhodování.

Co to je strukturované programování

Podle [1] a [2] se za strukturovaný považuje program, který je napsán jako posloupnost příkazů, přičemž příkazem se rozumí:

- výkonný příkaz (obr. 1a)
- podmíněný příkaz (výběr ze dvou možností podle výsledku testu) (obr. 1b)
- příkaz cyklu (tělo cyklu se provádí, dokud je splněna testovaná podmínka) (obr. 1c) .



Obr. 1

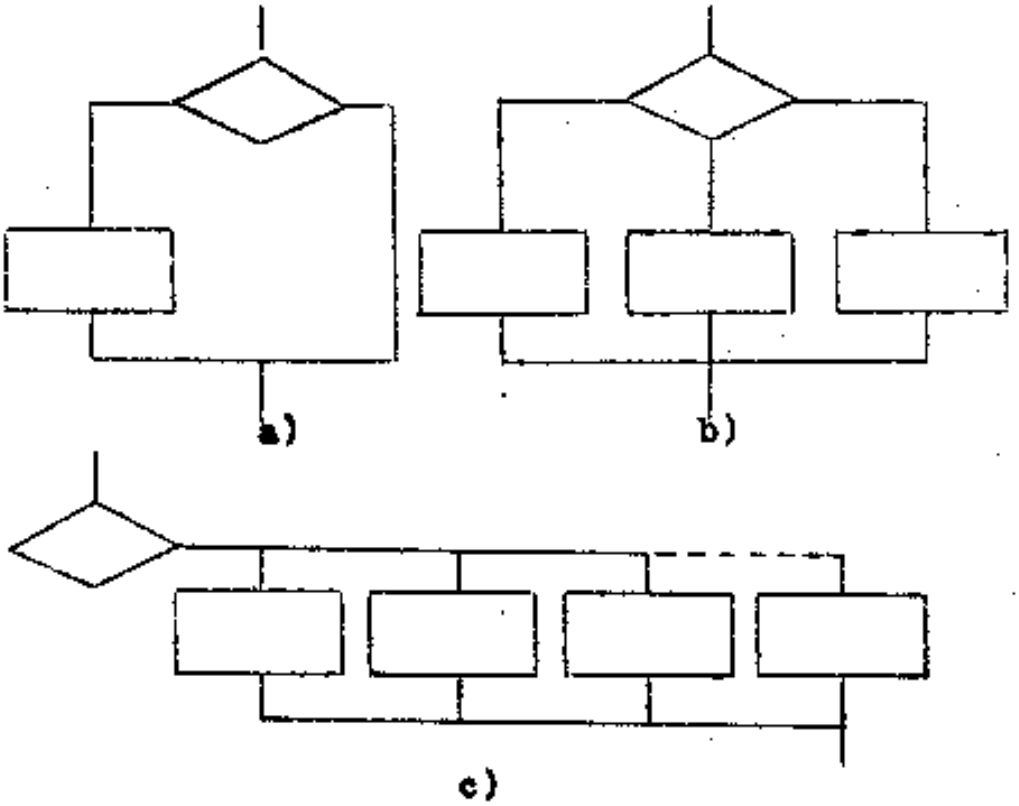
Výkonným příkazem přitom rozumíme buď přímo příkaz programovacího jazyka nebo příkaz k provedení posloupnosti příkazů podle výše uvedených pravidel.

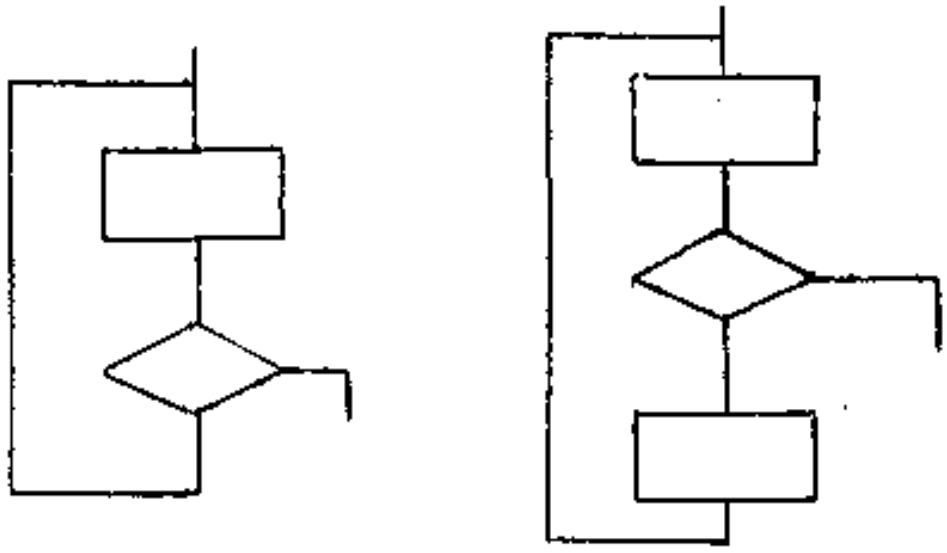
Strukturované programování se obvykle používá současně s metodou shora dolů. Programovat konkrétní úlohu začneme tak, že napíšeme jediný příkaz řešící úlohu. Tento příkaz podle výše uvedených zásad strukturovaného programování rozepíšeme jako posloupnost podrobnějších příkazů; každý takový příkaz opět rozpracujeme (postupně zjemňujeme) až do úrovně příkazů zvoleného programovacího jazyka.

Pro příkazy volíme vhodné názvy. To je myšlenkově obdobná činnost jako vymýšlení názvů odstavců v COBOLu, subrutin ve FORTRANu, procedur v ALGOLu či PASCALu, apod.

Z praktických důvodů povolujeme používání i dalších struktur, které lze jednoduše převést na výše povolené struktury, a to

- test nemusí být jen binární, obr. 2a, b, c
- cykl může mít test i jinde než před prvním příkazem, obr. 2 d, e





d) e)

Obr. 2

Lze programovat strukturovaně?

V přednáškách o strukturovaném programování bývá zvykem provést důkaz nebo aspoň na příkladu ilustrovat, že každý program, který se dá popsat vývojovým diagramem (s jedním vstupem a jedním výstupem z příkazu), lze zapsat též strukturovaným vývojovým diagramem [2], [3] . Kromě samotného faktu, že strukturovaně programovat lze, nedává zmíněný důkaz pro programátora v praxi žádnou použitelnou informaci. Kdybychom programovali libovolně a pak podle metody, kterou daný důkaz používá, převedli náš program do strukturovaného tvaru, dostali bychom většinou program nepřehledný, dlouhý a pomalu pracující. Jakékoli mechanické převody z nestrukturovaného do strukturovaného tvaru prováděné programátorem je třeba odmítnout. Programátor by neměl transformovat program do strukturovaného tvaru, nýbrž rovnou psát strukturovaně.

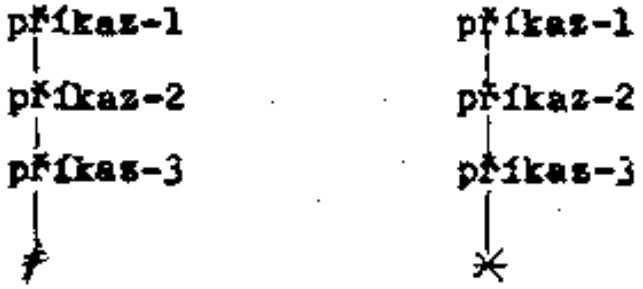
Při vytváření vývojového diagramu je obtížné používat jen povolené konstrukce. Kromě toho není z vývojového diagramu obvykle zřejmé, jak vznikal postupným zjemňováním, což je vhodné [4] , [5] . Zejména při pozdější údržbě je velké riziko zásahů porušujících strukturovanost.

Potřebujeme tedy metodu, která si vynutí strukturované řešení, neumožní použít nepovolené konstrukce, zachytí postup shora dolů a u které ani pozdější údržba nenaruší strukturovanost řešení. Tyto požadavky splňují různé metody, kde je program zapsán jako rozhodovací strom, viz např. [2] . Metoda, kterou zavedl R. Witty [5], [6] je navíc programátorskému myšlení velmi blízká.

SPR

Diagram pro strukturované řešení shora dolů musí umožňovat zápis posloupnosti příkazů, postupného zjemňování, rozhodování, cyklu a použití předem definovaných procedur.

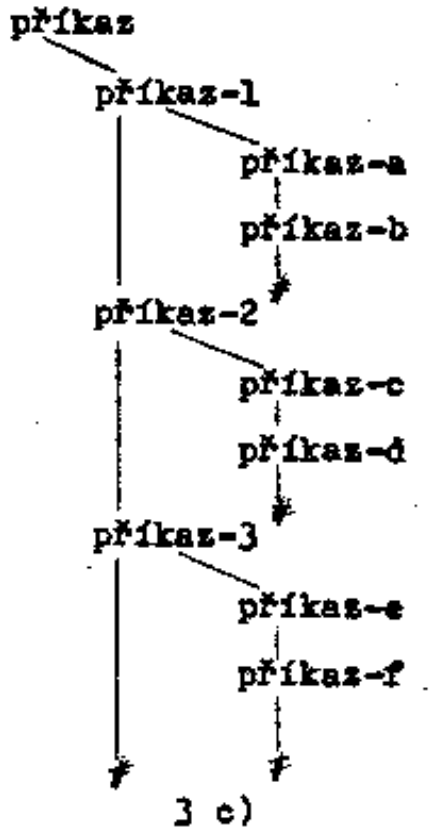
Příkazy v posloupnosti píšeme pod sebe a spojujeme svislou čarou (bez obdélníků z "klasických" vývojových diagramů). Každou posloupnost ukončíme znakem $\not$ nebo $\ast$ (vysvětlení níže) viz obr. 3a, b.



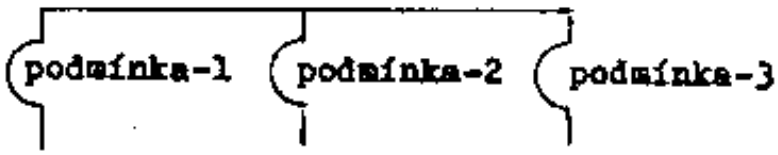
3 a) 3 b)

Od příkazu, který chceme rozepsat do větších podrobností, vedeme svislou čáru šikmo vpravo dolů, pod jejíž dolní konec zapíšeme podrobnější posloupnost příkazů. Tyto příkazy se mohou obdobně dále zjemňovat. Viz obr. 3c.

Podmínky zapisujeme k obloučku ve tvaru C (od slova condition-podmínka), při vícenásobném testu zapíšeme podmínky vedle sebe a obloučky vyznačující podmínku spojíme nahoře vodorovnou čarou, viz obr. 3 d. Podmínky vyhodnocujeme odleva doprava, provádění posloupnosti příkazů pokračuje dolů první větví se splněnou podmínkou. Pokud není splněna žádná podmínka, postupujeme tak, jak by v daném místě diagramu byl znak $\not$; obr. 4a a 4b jsou ekvivalentní.

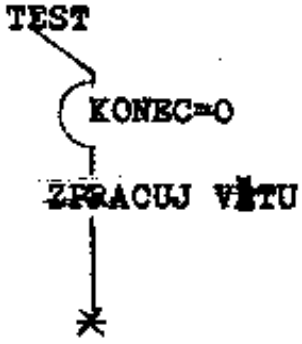


3 c)

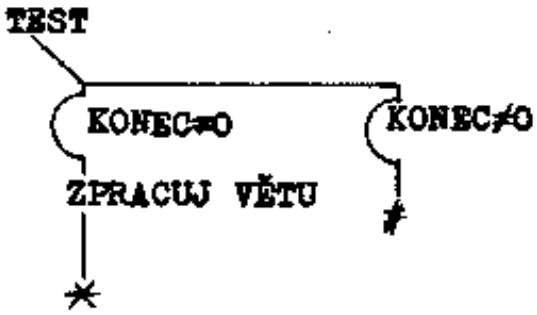


3 d)

3 e)

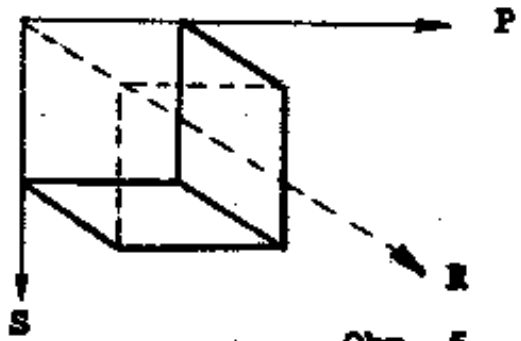


4 a)



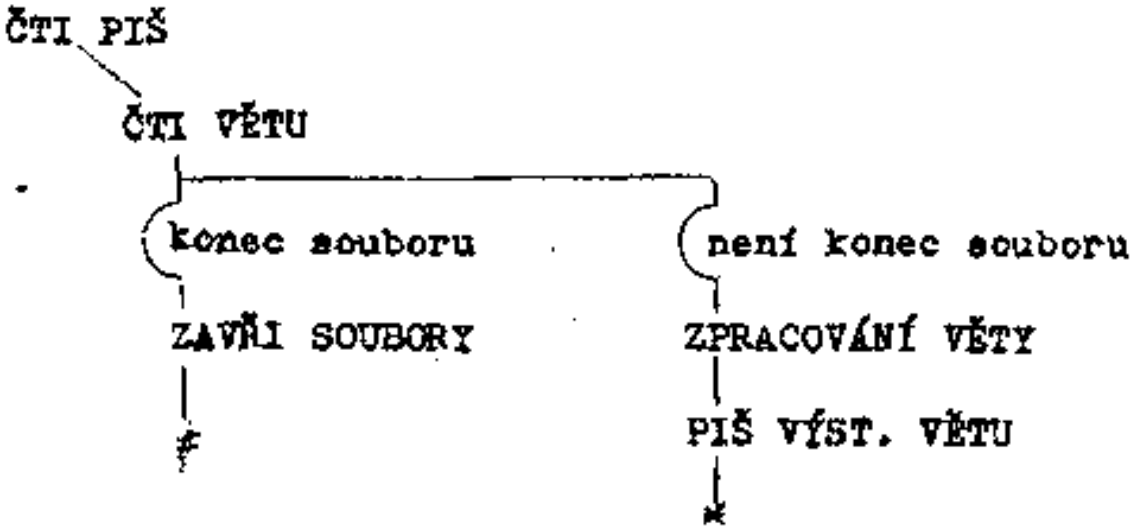
4 b)

Diagram SPR si můžeme představit jako dvourozměrnou projekci trojrozměrného diagramu, v němž na jednotlivých osách vynášíme posloupnost (S), paralelní alternativy při rozhodování (P) a zjemnování (R), viz obr. 5.



Obr. 5

Každá posloupnost příkazů (a také každá větev při rozhodování končí znakem ≠ kromě případů, kdy si přejeme, aby se celá posloupnost opakovala, což zajistíme znakem * viz obr. 6.



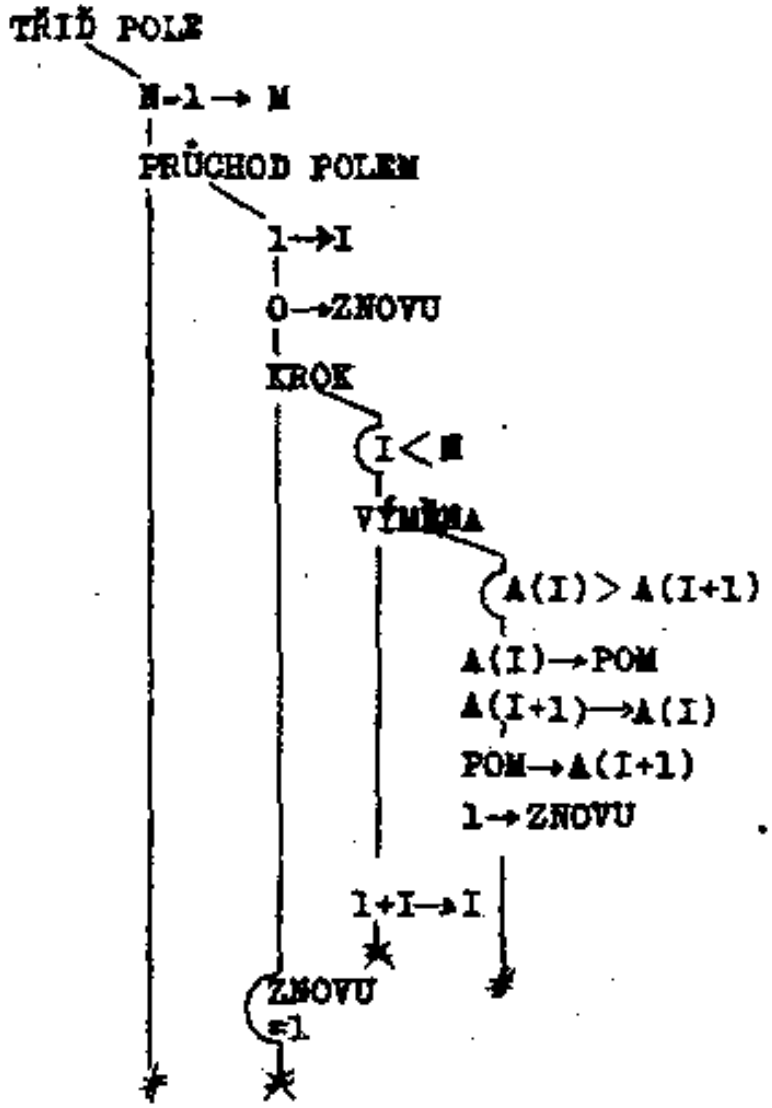
obr. 6

K příkazu, který dále nerozepisujeme, protože je definován jinde (procedura, subrutina), nakreslíme rovněž šikmou čáru, kterou na konci opatříme šipkou, viz obr. 3 e.

Příklad

Ukažme si použití metody SPR na jednoduchém školním příkladu.

Setřiďte prvky pole a v prvcích podle velikosti. Metodu zvolíme tu nejjednodušší. Pole budeme postupně procházet a nesprávně seřazené sousední prvky přehodíme. Průchody opakujeme, dokud neprojdeme bez výměny prvků, viz obr. 7



obr. 7

Automatizace kódování

Vzhledem k tomu, že se v metodě SPR téměř nepoužívají speciální značky, je poměrně snadné si představit uložení diagramu na vhodném médiu (diskový soubor) a s využitím editačních prostředků moderních operačních systémů i vytváření, doplňování, úpravy a opravy diagramu z interaktivních, nejlépe obrazovkových terminálů. Potom lze i diagramy tisknout tiskárnou počítače nebo kreslit grafickým výstupním zařízením. Lze i automatizovat kódování v programovacím jazyku, takže programátor by neměl zdrojový text a diagram, nýbrž jen diagram, viz [5] , [6] , [7]. V [5] a [6] uvádí R. Witty zkušenosti s automatizací diagramů pro kódování v assembleru a Fortranu.

Zhodnocení metody SPR

V oddělení programování VC ČKD PRAHA používají někteří pracovníci tyto diagramy již téměř půl druhého roku. Diagramy SPR jsme souběžně s "klasickými" vývojovými diagramy učili i v kursu pro programátory. Začátečníkům nečiní metoda SPR potíže. Problém je se změnou myšlení u zkušených programátorů a ve vztahu analytika a programátora (z nestrukturovaného zadání se strukturovaně programovat nedá).

Shrňme si výhody metody SPR:

- výsledný program je vždy strukturovaný;
- výsledné řešení se vždy získá metodou shora dolů;
- struktura programu i postupné zjemňování jsou v diagramu zachyceny;
- alternativy jsou nakresleny vedle sebe;
- nelze začít předčasně kódovat, dokud není jasná struktura programu;
- největší myšlenkové úsilí je třeba vynaložit na počátku práce, kdy se řeší struktura programu jako celku (u "klasických" vývojových diagramů v závěru, kdy se už programátor ve "struktuře" programu leckdy ani nevyzná);
- práci lze snadno rozdělit více programátorům (vedoucí týmu napíše nejvyšší úroveň/úrovně);
- programátor i vedoucí skupiny mohou snáze sledovat postup prací;
- komunikace s jinými programátory nebo s vedoucím týmu je snazší;
- údržba je snazší; chyby a úpravy se obvykle týkají nejnižších úrovní diagramu (navíc u "klasických" vývojových diagramů zásahy obvykle nerespektují dosavadní "strukturu" a logiku řešení);
- diagram SPR lze snadno uchovávat na počítačovém médiu, což umožní automatizaci kódování z diagramu i automatizaci některých částí dokumentace.

Nevýhody:

- diagram SPR je obvykle třeba kreslit na papír velkého formátu;

- ČSN pro dokumentaci připouští pouze "klasické" vývojové diagramy;
- výsledný program může být poněkud pomalejší a nepatrně větší, vzhledem k tomu, že nelze optimalizovat na úkor strukturovanosti.

Závěr

Ověřili jsme a v praxi vyzkoušeli metodu diagramů SPR. Tuto metodu doporučujeme k použití zejména pro programování ve vyšších programovacích jazycích. K úspěšnému používání je ovšem třeba změnit způsob práce a myšlení programátorů i analytiků.

Literatura

1 Hořejš, J.: Strukturované programování. Sofsem '74.

2 Linger, R.C. - Mills, H.D.: On the Development of Large Reliable Programs, Current Trends in Programming Methodology (editor Yeh R.T.), Vol.I, s.120-139.

3 Kralovič, E.: Použití strukturovaných prvků při programování v jazyku COBOL. Sborník přednášek ze 6. semináře Používání jazyka COBOL, Pardubice 1979.

4 Wulf, W.A.: Languages and Structured Programs, Current Trends in Programming Methodology (editor Yeh R.T.), Vol. I., s. 33-60.

5 Witty, R.W.: The design and Construction of Hierarchially Structured Software, Pragmatic Programming & Sensible Software, s. 361-388, 1978.

6 Witty, R.W.: Dimensional Flowcharting, Software-Practice and Experience, Vol. 1, s. 515-583, 1977.

7 Nichtburger, E.: Třírozměrné vývojové diagramy typu SPR, ASŘ-bulletin Inorga 1979, č. 2, str. 78-80.

8 Nichtburger, E.: Metoda vývojových diagramů typu SPR, vyjde ve sborníku Semináře o zkušenostech a novinkách v programovém vybavení počítačů ODRA, 1980.