Ing. Pavel Ligenza
Důl J. Fučík, k.p., Petřvald

ZKUŠENOSTI S VYUŽÍVÁNÍM NĚKTERÝCH MODERNÍCH METOD PROGRAMOVÁNÍ PRO MINIPOČÍTAČE

Dále uvedené poznatky byly získány na výpočetním středisku, vybaveném počítači ADT 4100, bez externích pamětí. Vzhledem na toto omezení, dané technickými možnostmi počítače, nemůžeme aplikovat poznatky pracovníků od středních a velkých počítačů bez určitého přizpůsobení. Pro racionalizaci programátorské práce jsme použili tyto metody: modulární stavbu programů, řízení programů externě definovanými prvky, programování metodou SPR a zavedení programovacích standartů. Dále uvádím, že programujeme v jazyku BASIC FORTRAN ADT.

1. Modulární stavba programů (1).

Již před zahájením práce na modulu rozhodujeme, zda půjde o modul pro jednorázové nebo opakované použití. Jednorázový modul má název shodný s programem, doplněný rozlišovacím písmenem A - Z, opakovaný modul má název odvozený z jeho činnosti.

Jednorázové moduly jsou přes top modul spřaženy společným COMMON blokem, vyžaduje-li to lepší srozumitelnost programu, mají navíc i vnější spřaženost s top modulem. Soudržnost těchto modulů bývá funkční nebo procedurální, vyjímečně

jiná. Snažíme se, abychom jednorázové moduly nepoužívali do nižší úrovně, než top -2. Společný COMMON je výhodou při pozdějších změnách programů, neboť výpočtová pole jsou přístupná ze všech modulů vyšší úrovně. Nevýhodou je možnost neúmyslného přepsání pole.

U modulů pro opakované použití dodržujeme vnější spřaženost a funkční soudržnost. U modulů, určených k zařazení do systémové knihovny, povolujeme navíc spřaženost řízením, a to pro úsporu místa v systémové knihovně, jejíž velikost je omezena. Modul pro knihovnu musí obsáhnout více variant základní činnosti (nejlépe všechny). Volba varianty se provede parametrem.

Potřebujeme-li modul, pracující s dvourozměrným polem, navrhnout pro opakované použití, je třeba převést jeho spřaženost přes datovou základnu na spřaženost vnější. Vzhledem k tomu, že při předávání jednorozměrného pole přes parametr jde de facto o spřaženost vnější (předává se jen počáteční adresa pole na rozdíl od pole dvourozměrného, kdy se předává jednak počáteční adresa a jednak první dimense), stačí toto pole převést na jednorozměrné. V modulu (subroutině) pak dostačuje udat popis DIMENSION POLE (1). Abychom v modulu mohli pracovat s polem jako s dvourozměrným aspoň fiktivně, používáme mapovací funkci MAP (IND1, IND2, IDIM)= (IND2-1) * IDIM1 + IND1, kde IND1, IND2, IDIM můžeme předat přes parametry. Vzhledem na přípustné typy aritmetických výrazů vyjádříme jednorozměrný INDEX = MAP (IND1, IND2, IDIM) a dále VYSL = POLE (INDEX).

2. Řízení programu externě definovanými prvky (2).

Kromě způsobů využití popsaných v literatuře (2) umožňuje nám tato metoda zjednodušit tiskové moduly, nahradit

nepřehledné sekvence instrukcí cyklem a uplatnit t.zv. programování "bez konstant".

Basic Fortran ADT nepřipouští proměnný FORMAT u IO instrukcí. Proto při klasickém způsobu programování je tiskový modul sekvencí instrukcí WRITE a FORMAT a to pro každý řádek jiných. Použití předem naplněného textového pole umožní řešit tiskový modul podle vývojového diagramu z obr. 1.

Sekvenci rozhodovacích bloků, srovnávajících postupně určitou proměnnou se zcela nepravidelnou řadou čísel (na př. zařazení porubu do skupin dle velikosti mocnosti sloje) lze nahradit jednoduchým cyklem dle obr.2, využívajícího předem naplněné pomocné pole.

Při dodatečných zásazích do programu, ať již jsou vyvolané změnou vnějších podmínek, nebo přílišným optimismem pracovníka, který zpracoval úvodní studii, bývá nejobtížnější opravit beze zbytku všechny konstanty v cyklech, rozhodovacích blocích, skutečných parametrech subroutin a p. Došli jsme k závěru, že při změnách, přestavbách či opravách programů je výhodné mít možnost provést tyto změny centrálně. Používáme proto více méně důsledně místo těchto konstant prvky polí nebo proměnné, s předem do nich uložené hodnoty.

Naplnění řídících a textových matic, polí a proměnných provádíme v samostatném modulu, který je buď připojen k vlastnímu programu a jeho vyvolání je řízeno z pultu počítače, nebo je zpracován jako samostatný program. V obou případech je konečným produktem děrná páska v binárním kódu s absolutními adresami, které se do paměti načítá stejně jako program. Její umístění v paměti počítače je přesně definováno jako část pole COMMON umístěná hned za operačním systémem BOS. Toto umístění je dosaženo tím, že příslušné matice, pole a proměnné jsou umístěny na formálním konci popisu COMMON. Proto je tato řídící páska vázána na určitou versi operačního systému.

Pro usnadnění tvorby modulu pro generování řídící binární pásky jsme zpracovali soubory subroutin UVOD a GENES, které po doplnění jednoduchým top modulem zajistí načtení, tisk, eventuelní opravu a vyděrování opravené vstupní pásky pro jednotlivé řídící matice a pole a také vyděrování souhrné řídící binární pásky.

3. Programování metodou SPR. (3).

V naší modifikaci zdůrazňujeme princip postupného zjemňování programu a to již ve stadiu studie (projektu) programu, stejně jako rozčlenění programu do jednotlivých úrovní (top, top-1, ..., základní modul). K podstatným změnám došlo ve formálním zápisu, kdy přirozeným výběrem vznikl hybrid mezi klasickým postupovým diagramem a diagramem Wittyho.

Diagram je kreslen na jednotlivých listech a to tak, že každá sada listů znázorňuje vlastně jeden sloupec SPR diagramu. Dodržujeme zásadu "jeden modul-jeden list", samozřejmě s výjimkou posledního sloupce, který je representován zápisem programu na programovacích formulářích. První list tak de facto představuje top modul (1.sloupec SPR diagramu), druhá sada listů pak top-1 moduly, atd. U základních (knihovních) modulů se předpokládá jejich znalost a nerozepisují se. Pokud jde o formální úpravu, pak každý příkaz, který není samostatnou instrukcí obsahuje název (popis), označení úrovně s jméno (název subroutiny). Rozpis každého příkazu (modulu) začíná uvedením jeho nacionálií v levém horním rohu (název, úroveň, jméno) a končí příkazem RETURN nebo STOP v pravém dolním rohu. Zápis jednotlivých příkazů odpovídá zvyklostem klasického postupového diagramu, s výjimkou přepínače (SWITCH), který se zapisuje způsobem, znázorněným na obr. 4.

4. Programovací standerty.

Zásady programování jsou shrnuty v písemnosti nazvané Programovací standarty 81, které obsahují: povolené programovací jazyky (Fortran, Assembler), předpis pro formální úpravu programů (standartní názvy top-1 modulů, formální úprava modulu, ostatní ustanovení), standarty pro vývojový diagram (popis formální úpravy, postup vypracování, zásada "1 modul - 1 list" a výjimky, odchylky od ČSN), popis stavby modulu (definice modulu, povinné vybavení modulu, zápisy do knihovny, výjimky), top-1 moduly (pro jednotlivé moduly: možné základní moduly pro jednotlivé funkce, dohodnutý způsob práce, ostatní dohodnuté náležitosti), reservace tlačítek pro ruční řízení programu, nestandartní použití Fortranu na ADT, popis jednotlivých versí operačního systému BOS a přidělena čísla periferií, knihovna základních modulů (soupis modulů pro ladění programů a modulů ostatních, popis jednotlivých modulů, odkaz na veřejně přístupný exemplář zdrojových výpisů). Programovací standarty jsou každoročně aktualisovány.

5. Příklady praktického použití.

Jako příklad použití uvádím řešení programu s průběžně proměnným zadáním. Úkolem bylo zajistit denní sledování spotřeby materiálu v Kčs a to jak denně, tak v součtu od počátku měsíce, za podnik, závody, hospodářské a nákladové středisko a pracoviště. Údaje o denní spotřebě materiálu jsou výpočetnímu středisku dodávány ve formě děrné pásky, vznikající při odpisu materiálu z karet na strojích Ascota. Členění na pracoviště se vyskytuje jen u některých nákladových středisek. Pracoviště (rubání) jako takové jsou nestabilní, vznikají a zanikají i během měsíce a jejich vznik a zánik nelze ve všech případech předvídat. I stabilita hospodářských a nákladových středisek je pouze relativní, jejich změny při-

cházejí k začátku roku.

Problém byl řešen metodou řízení programu externě definovanými prvky, kdy řídící páska je produktem samostatného programu. Vlastní program se tak stal zcela nezávislý na složení jednotlivých sumarisačních úrovní. S řídící páskou lze ve značném rozsahu manipulovat pomocí dalších programů, řízených údaji, které dodá uživatel dle potřeby na vstupních formulářích.

Podle základního vstupního formuláře, obsahujícího v řádcích vždy číslo závodu, hospodářského a nákladového střediska, rozlišení, zda jde o nákladové či hospodářské středisko, příslušný název a počet pracovišť a formuláře pro pracoviště, kde ke každému nákladovému středisku, složenému z pracovišť jsou přiřazena jejich čísla, se v paměti počítače vytvoří základní pole, pole hospodářských středisek, spojovací pole a pole pracovišť (obr. 5). Tato pole jsou vyděrována jako binární páska. Současně počítač tiskne formulář pro vstup plánu nákladů. Jeho skladba odpovídá skladbě příslušných řídících polí. Plány se uloží do reservovaných polí řídící binárky a tím vznikne její definitivní podoba.

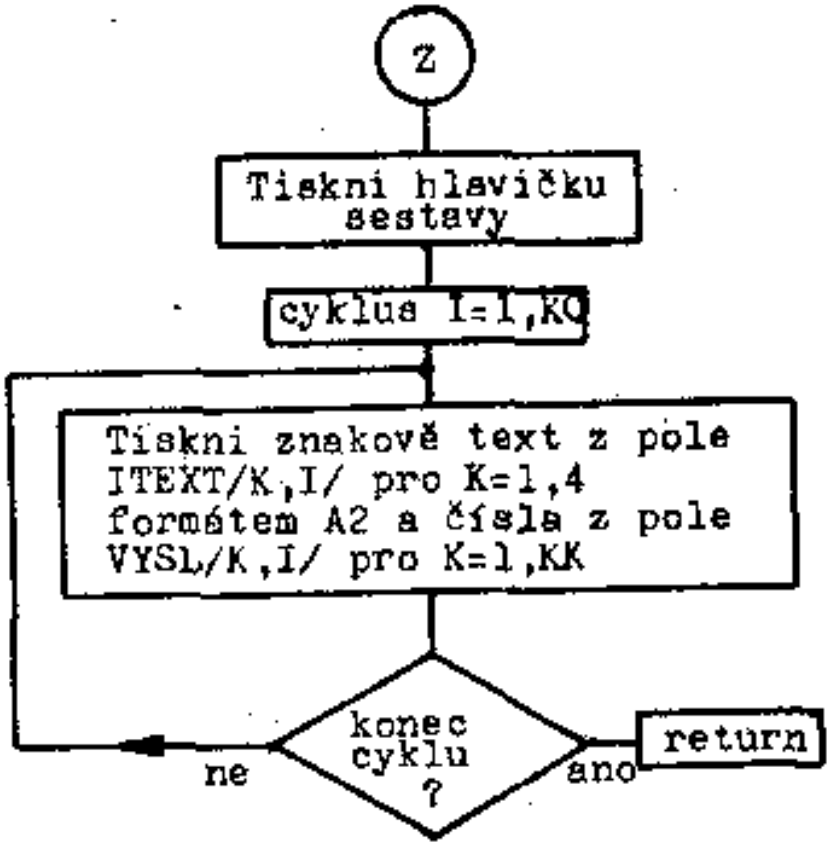
Dojde-li během měsíce ke vzniku nového pracoviště, změní se číslo záložního pracoviště, které jsou u všech nákladových středisek. U pracovišť tohoto nákladového střediska lze měnit plánované náklady. Při aktualizaci řídící binárky na počátku měsíce jsou možné libovolné změny počtu pracovišť.

Řízení programu pomocí řídících polí je záležitost rutinní a proto pouze jako příklad uvádím způsob sumarizace materiálové položky (obr. 6).

Na proměnné XKONT, XKČS a XPRAC jsou uloženy hodnoty přečtené z děrné pásky, vydérované na stroji Ascota současně s odpisem materiálu ze skladové karty, a to číslo konta, součin odebraného množství a ceny za jednotku a číslo pracoviště. Z konta lze odvodit jak číslo závodu, tak i číslo hospodářského a nákladového střediska. Není-li v poli kont RNSKO nalezeno konto XKONT, znamená to, že na výdejce bylo udáno neexistující konto a položka je započtena na chybové konto. V opačném případě se náklady na položku sumarisují pro podnik (POD(2)), nákladové středisko (RNSUM(IND)) a po nalezení příslušného indexu hospodářského střediska INHS v poli INS také pro hospodářské středisko. Po nalezení příslušného indexu INZAV pro závod v poli IHS také pro závod. Následuje vyhledání indexu INSP pro pracoviště. Je-li tento index nulový, znamená to, že příslušné nákladové středisko není rozčleněno na pracoviště, a proto je další část programu přeskočena. V opačném případě jsou z pole IHSPR určeny indexy, mezi kterými mohou ležet příslušná čísla pracoviště. Není-li v těchto mezích číslo pracoviště nalezeno, znamená to chybu ve vyplnění výdejky a sumarizaci na chybové konto. Jinak jsou náklady na položku přičteny na příslušné pracoviště (TPRAC(IND)).

Literatura

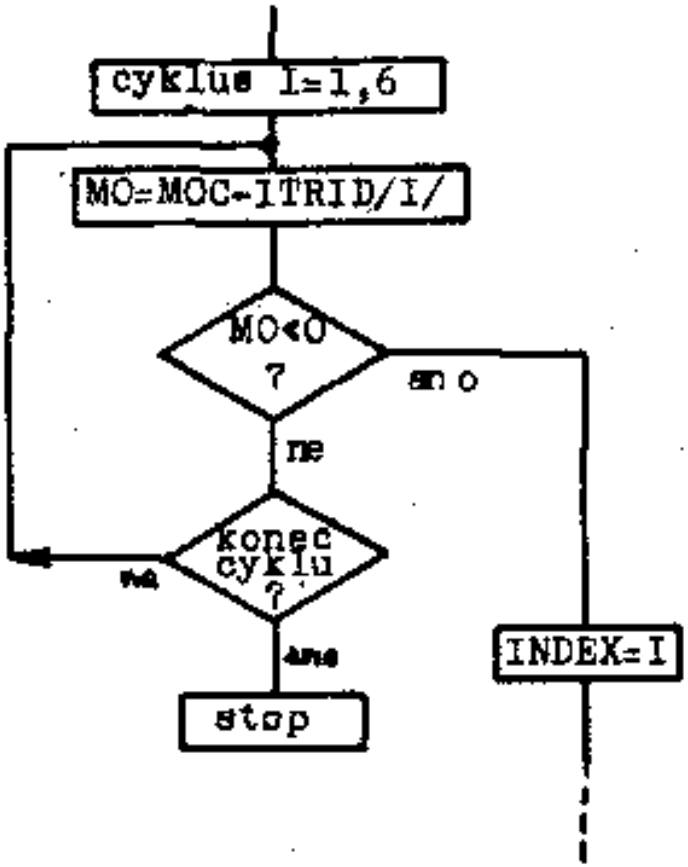
1 Čimbura V., Tvrdík J.: Problémy návrhu modulárních programů
2 Věžník M.: Řízení programů externě definovanými prvky
3 Nichtburger E.: Strukturované programování metodou SPR
Všechny 3 stati jsou uveřejněny ve sborníku PROGRAMOVÁNÍ 80



	1	2	3	4
1	C H	O D	B Y	
2	P R	O R	A Z	K Y
3	P R	E K	O P	Y
4	I V	V R		
5	G P	P		
6	V O	K D		
...				
KC	C E	L K	E M	

Popis pole ITEXT (4,.)

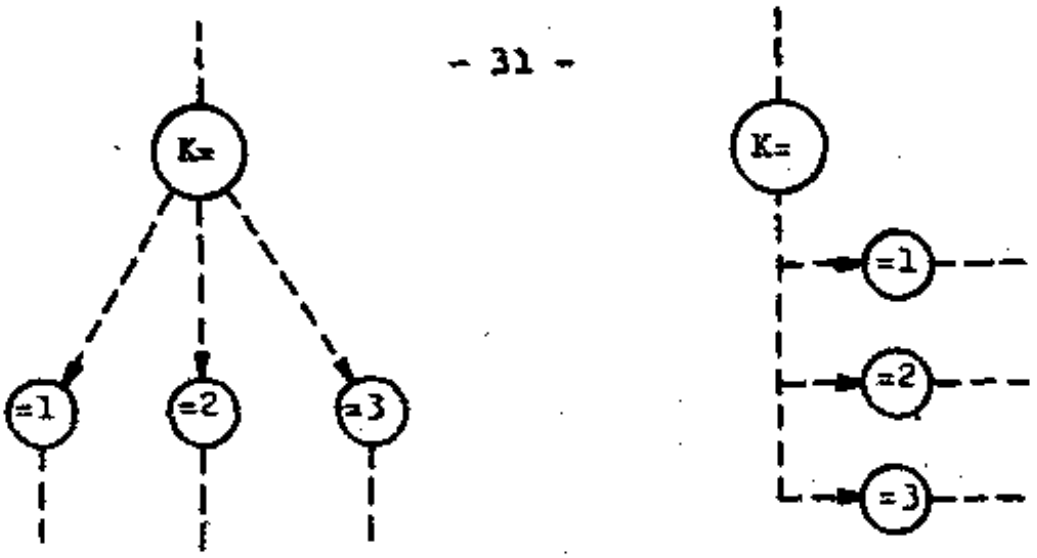
OBR. 1



Popis pole ITRID (6)

1	50
2	80
3	120
4	180
5	260
6	800

OBR. 2



Klasický zápis

Upravený zápis

OBR. 4

Popis řídících polí pro progr.Sledování spotřeby materiálu.

Z á k l a d n í p o l e :

konto A RNSKO(x)	sum.buň. RNSUM(x)	plán RPLNS(x)	odkaz do pole RNSUM a IHS INS(x,1)	odkaz do pole RNSPR a IHSPR INS(x,2)	název nákl.s. IHSPR(x,3) až IHSPR(x,8)

Pole hospodářských středisek :

sam.buňka RHSPR(x)	plán RPLHS(x)	konto A/100. IHS(x,1)	odkaz do pole ZAV IHS(x,2)	název hosp. střediska IHS(x,3) až IHS(x,8)

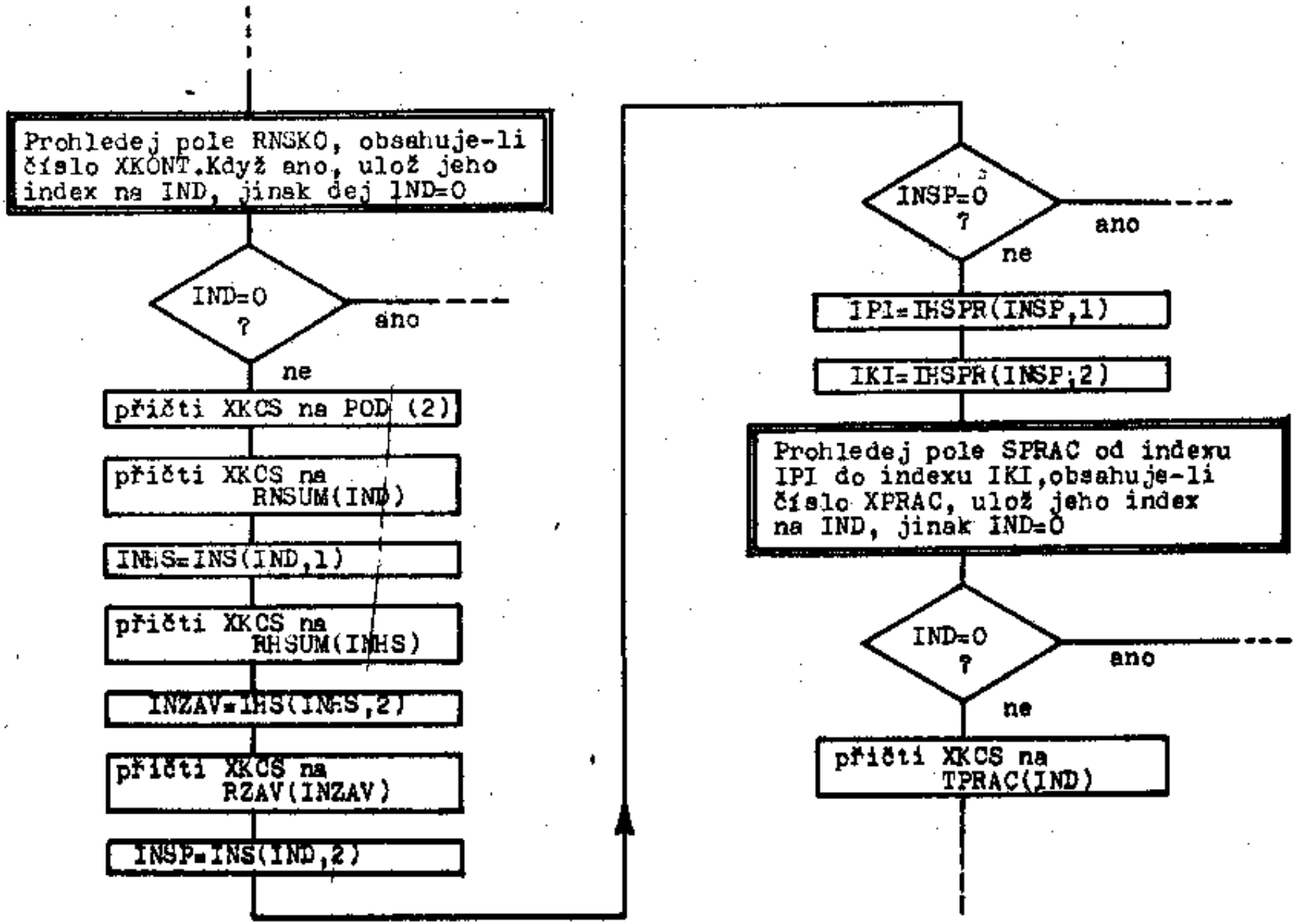
S p o j o v a c í p o l e :

konto A RHSPR(x)	počáteční index v poli RPRAC IHSPR(x,1)	koncový index v poli RPRAC IHSPR(x,2)

P o l e p r a c o v i š ť :

číslo pracoviště SPRAC(x)	konto A RPRAC(x,1)	sum.buňka TPRAC(x)	plán RPLPR(x)

OBR. 5



XKONT ... konto s výdejky
XKCS .... čerpané náklady
XPRAC ... č.prac.z výdejky
RZAV(i).. sumarizace závodů
POD(2) ... sumarizace podniku

OBR. 6