OD PROBLÉMOV KU PROGRAMOM I

JOZEF HVORECKÝ, JOZEF KELEMEN, Bratislava

Jedni povedia, že som si to všetko vymyslel, že to nie je tá pravá náuka; iní zas, že som povedal len to, čo už každý vie.

Anatole France

V článku sa zaoberáme analýzou procesov tvorby postupov riešení problémov v rozličných problémových prostrediach. Cieľom analýzy je nájsť spoločné črty týchto procesov a formulovať tie metódy tvorby postupov riešení, ktoré nie sú primárne viazané na jednotlivé problémové prostredia, ale sú v istom zmysle univerzálne použiteľné. Znalosť takýchto postupov môže byť, podľa nášho názoru, výhodná pri vytváraní programov na riešenie problémov pre počítače. Článok voľne naväzuje na niektoré myšlienky, načrtnuté v našej práci [7].

1. Úvod

Človek sa oddávna snaží odhaliť postupy, ktoré by mu umožnili ľahšie, rýchlejšie a pritom spoľahlivo vytvárať postupy riešení vytýčených problémov. Temer všetci význační myslitelia všetkých historických období sa zamýšľali nad existenciou „univerzálnej" metódy riešenia problémov.

Už starogrécky matematik Pappos (žil pravdepodobne v 3. storočí pred n. l.) píše vo svojom diele *Collectio* o potrebe vzniku vednej oblasti, ktorou by sa mali zaoberať tí, ktorí si po osvojení základov matematiky chcú osvojiť i schopnosť samostatne riešiť matematické problémy.

„Keď som v mladosti počul o niektorých objavoch, pokúšal som sa vykonať ich sám, bez toho, aby som čítal diela ich autorov. Pri tomto mojom počínaní som postupne badal, že moje myslenie sa riadi určitými *pravidlami.*" píše R. Descartes (1596—1650).

G. W. Leibniz (1646—1716) sa údajne pripravoval napísať knihu o „ars inveniendum" a v mnohých jeho dielach sa dajú nájsť zmienky o mimoriadnom význame, ktorý pripisoval spôsobom nachádzania riešení problémov. *Cesty*, ktoré k objavom vedú, sú, podľa neho, niekedy cennejšie ako samotné objavy.

Veľkú pozornosť *heuristike*, čiže „umeniu riešiť problémy", venoval i B. Bolzano (1781—1848). Nasledujúce jeho slová by sme i my radi odporúčali do pozornosti čitateľov tohoto článku. *„Vôbec si nemyslím, že podávam také metódy myslenia, ktoré by rozumný človek oddávna nepoznal, vôbec nesľubujem, že čitateľ v súvislosti s týmito metódami nájde tu čosi úplne nové. Ale neľutujem námahu jasnými slovami opísať metódy a pravidlá myslenia, ktoré každý rozumný človek používa, aj keď vo väčšine prípadov neuvedomele."*

Azda najznámejším (najmä medzi matematikmi a učiteľmi matematiky) predstaviteľom podobných snáh je v súčasnosti známy matematik a pedagóg G. Pólya, ktorého diela [10, 11, 12] sa stali jedným z nosných pilierov nielen modernizačného hnutia v metodike matematiky, ale poslúžili i ako východisko alebo ako bohaté žriedlo ponaučenia pri serióznych výskumoch metód riešenia problémov.

Silným impulzom pre výskum v oblasti tvorby postupov riešení problémov bolo objavenie *výpočtovej techniky* a nutnosť *programovať* postupy práce počítačov. *Program* na základe ktorého je počítač schopný vyriešiť problém nie je totiž nič iné, než *postup riešenia* toho problému, zapísaný v špeciálnom jazyku, ktorému hovoríme *programovací jazyk*. Kvôli presnosti je potrebné poznamenať, že postupy riešení problémov, vykonávanie ktorých chceme prenechať počítačom, musia vyhovovať určitým podmienkam (konečnosť, jednoznačnosť, efektívnosť). Splnenie týchto podmienok znamená, že daný postup riešenia môžeme považovať za *algoritmus* (bližšie o tom pozri napríklad v [7]). Ak podmienkam kladeným na algoritmy, daný postup riešenia v plnej miere nevyhovuje, nemusí to ešte znamenať, že vykonávanie daného postupu nemožno prenechať počítaču. V takomto prípade však musíme byť pripravení na to, že počítač nemusí vždy nájsť riešenie daného problému, resp. nemusí nájsť riešenie každého problému z danej triedy (ktorá obyčajne v takýchto prípadoch nie je dostatočne presne vymedzená). O tejto problematike možno nájsť podrobnejšiu zmienku v [8].

Počítač, ak má vhodný program, je schopný spracovávať také množstvá údajov za pomerne krátky čas, na spracovanie akých by človeku nestačil niekedy ani celý život. Z tejto skutočnosti vyplýva v niektorých prípadoch veľmi nepríjemný fakt, že človek nie je schopný skontrolovať *správnosť riešenia,* ktoré našiel počítač. Jednou z ciest, ako upevniť svoju vieru v to, že riešenie je správne, je byť presvedčený o tom, že postup riešenia je správny. Ak však chceme vytvárať správne postupy riešenia problémov, mali by sme sa snažiť vytvárať ich spôsobom, ktorý v maximálnej miere eliminuje možnosti omylov. Snahy o nachádzanie takýchto postupov sú v súčasnosti späté hlavne s menom E. Dijkstra [2, 4].

Bolo by však nerealistické očakávať, že sa nám v dohľadnom čase podarí stanoviť také spôsoby tvorby postupov riešení problémov, ktoré by zaručovali ich bezchybnosť. Preto je dôležitá i otázka racionálneho nachádzania chýb a ich odstraňovania. Problematikou chýb pri tvorbe postupov riešení a v samotných postupoch sa začali odborníci zaoberať iba nedávno (prvé nám známe seriózne úvahy v tomto smere sú späté s menom G. J. Sussman [13]). Problematika je dodnes iba málo prebádaná, hoci kus zaujímavej a podnetnej práce v tomto smere bol už vykonaný. Práce I. P. Goldsteina a M. L. Millera [6, 9], dvoch priekopníkov v tejto oblasti, poslúžili i ako východisko a mohutný inšpiračný zdroj pri písaní tohoto článku. Zaoberajú sa v nich nachádzaním postupov riešení problémov samotným počítačom. My sme sa snažili ich výsledky aplikovať na človeka a miestami (napr. v časti 4.2) doplniť.

V tomto článku budeme ilustrovať naše úvahy o spôsoboch tvorby postupov riešení problémov predovšetkým na príkladoch z matematiky. Jazyk, v ktorom budeme postupy riešení formulovať, bude pripomínať programovacie jazyky. Sme však presvedčení o tom, že podobné spôsoby tvorby používa človek pri riešení drvivej väčšiny svojich problémov, či sa jedná o problémy každodenného života alebo o problémy vedecké. Iba jazyk, v ktorom sa postupy riešenia formulujú a problémové prostredia sú od oblasti k oblasti rôzne.

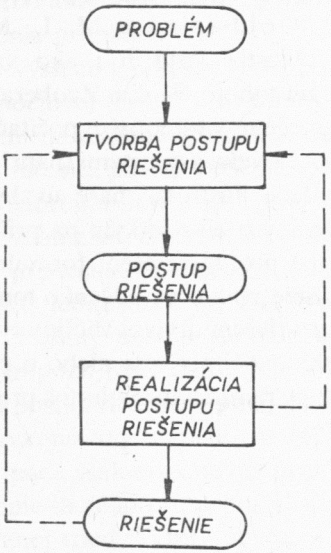
2. O problémoch

Vymedziť význam natoľko všeobecného pojmu akým je *problém,* je veľmi ťažké, ak nie nemožné. Vymedzenia, ktoré existujú (niektoré z nich sú uvedené napríklad v [5]), sa v mnohom vzájomne líšia, určite ich však spája aspoň jedna spoločná črta: možno proti nim vzniesť všetky námietky, ktorými obvykle napádame hmlisté definície. Ospravedlňujeme sa čitateľovi za to, že túto nepríjemnú vlastnosť bude mať i definícia, ktorú uvedieme my.

Problémom budeme nazývať stav, kedy pociťujeme potrebu doplniť existujúci systém svojich poznatkov o nový poznatok — *riešenie problému.*

Postup, ktorým obdržíme z pôvodného systému poznatkov riešenie problému, nazývame *postupom riešenia problému.*

Proces, ktorého cieľom je vytvoriť pre daný problém postup vedúci k jeho riešeniu, nazývame *procesom tvorby* postupu riešenia.



Obr. 1. Cesta od problému k jeho riešeniu

Ak teraz urobíme analýzu cesty, vedúcej od problémov k ich riešeniam zistíme, že táto cesta má nasledujúce dôležité etapy a míľniky: prvým míľnikom, ktorým sa cesta začína, je problém; prvá dôležitá etapa je tvorba postupu riešenia problému, končiaca druhým míľnikom — postupom riešenia; druhá etapa spočíva v *realizácii* postupu riešenia a končí posledným míľnikom — riešením problému. Schematicky znázorňuje opísanú cestu *obr. 1.*

K tomu, aby sme mohli začať s tvorbou postupu riešenia daného problému, potrebujeme mať problém najprv *sformulovaný,* t. j. musíme mať zreteľne stanovené, *čo* sa chceme dozvedieť. Tvorbu postupu riešenia potom môžeme chápať ako proces, meniaci takúto tzv. *deklaratívnu* formuláciu problému v *procedurálne* formulovaný postup riešenia, t. j. v postup, určujúci *ako* získať na základe poznatkov, ktoré máme, nový poznatok, predstavujúci riešenie.

V nasledujúcich častiach tohoto článku sa budeme zaoberať procesom tvorby postupov riešení, analýzou jednotlivých spôsobov tvorby a niektorými dôsledkami, ktoré vyplynú z našej analýzy pre vytváranie programov.

3. Tvorba postupov riešení

Keď človek rieši problém, nejakým spôsobom vždy reštrukturalizuje (rozširuje alebo iným spôsobom mení) systém poznatkov, ktorým disponuje. Dokonca *myslenie* sa v mnohých súčasných definíciách chápe ako schopnosť reštrukturalizácie systému poznatkov z hľadiska subjektu kvalitatívne novým spôsobom (porovnaj s [1]). Súčasné výskumy ukazujú (bližšie pozri [8]), že pri myslení a teda i pri riešení problémov využíva človek poznatky dvojakého druhu: *poznatky týkajúce sa problému,* ktorý sa chystá riešiť (pri riešení napr. matematických problémov sú to poznatky z matematiky) a *poznatky vyjadrujúce spôsoby tvorby* postupov riešení problémov, ktoré obyčajne človek nie je schopný v plnej šírke a presne slovne ani vyjadriť (takéhoto druhu sú napr. poznatky, ktoré nám umožňujú „vymyslieť" dôkazy matematických tvrdení).

Predmetom nášho ďalšieho výkladu budú poznatky druhého spomínaného druhu, teda tie poznatky, ktoré sú použiteľné pri tvorbe postupov

riešení problémov bez ohľadu na konkrétny problém, ktorého riešenie hľadáme.

Proces tvorby postupov riešení problémov pozostáva z dvoch základných činností: z *plánovania* postupu riešenia a z *odstraňovania chýb*, ktorých sme sa v etape plánovania mohli dopustiť. Pritom odstraňovanie chýb nie je nutnou súčasťou tvorby. Vykonávame ho iba v prípade (mimochodom, veľmi často sa vyskytujúcom), kedy sme sa pri plánovaní dopustili chyby.

4. Plánovanie

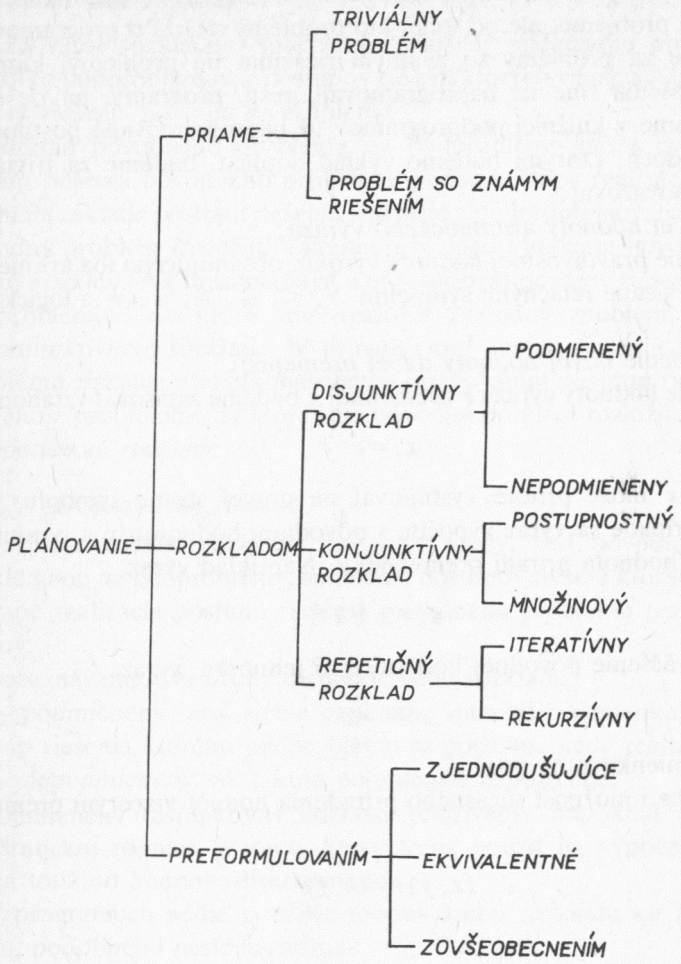
Existujú rôzne spôsoby, ktorými možno deklaratívne formulované problémy prevádzať v procedurálne formulované postupy ich riešení. V procese plánovania sa vykonáva postupná voľba takých spôsobov, ktoré umožňujú tento prevod vykonať. Na *obr. 2* uvádzame stručnú taxonómiu tých spôsobov plánovania, ktorými sa budeme v ďalšom zaoberať. Okrem vlastnej introspekcie sme pri jej zostavení čerpali z prác [6] a [9].

Ako je vidieť i z *obr. 2*, pri plánovaní máme možnosť voľby spomedzi troch spôsobov plánovania: priameho, plánovania rozkladom a preformulovaním.

Spôsob *priameho plánovania* možno použiť vtedy, keď postup riešenia daného problému plynie bezprostredne z deklaratívnej formulácie problému. *Plánovanie rozkladom* je v istom zmysle protipólom priameho plánovania. Uplatňujeme ho vtedy, keď je daný problém natoľko zložitý, že k tomu, aby sme ho mohli začať riešiť, je potrebné ho najprv rozložiť na niekoľko podproblémov. *Preformulovanie* problému spočíva v zámene pôvodného problému iným problémom, z postupu riešenia ktorého vieme získať postup riešenia pôvodného problému.

4.1 Priame plánovanie

K priamemu plánovaniu postupu riešenia problému sa uchyľujeme v tom prípade, keď daný problém identifikujeme ako triviálny, alebo ako problém s nám už známym postupom riešenia.



Obr. 2. Taxonómia plánovania

Triviálnym problémom rozumieme taký problém, z formulácie ktorého riamo vyplýva riešenie. Triviálnym je napríklad problém nájdenia maxi-nálneho prvku jednoprvkovej množiny.

Problémy so známym postupom riešenia sú tie problémy, ktorých ostupy riešení už nemusíme tvoriť, pretože sme sa to už naučili. Teda to,

či je alebo nie je problém problémom so známym postupom riešenia, nezávisí od problému, ale od toho, kto problémy rieši. Pri programovaní považujeme za problémy so známym riešením tie problémy, ktorých postupy riešenia sme už naprogramovali, resp. programy, na riešenie ktorých máme v knižnici podprogramov už naprogramované postupy.

V príkladoch, ktorými budeme výklad dopĺňať, budeme za triviálne problémy považovať:

— *výpočet hodnoty aritmetického výrazu;*

— *zistenie pravdivostnej hodnoty výroku,* obsahujúceho iba aritmetické výrazy viazané relačnými symbolmi $>$, $<$, $\geqq$, $\leqq$, $=$, $\neq$ a logickými spojkami $\wedge$, $\vee$, $\neg$;

— *priradenie novej hodnoty danej premennej.*

Priradenie hodnoty výrazu e premennej x budeme zapisovať vzťahom

$$x := e$$

Premenná x môže pritom vystupovať na pravej strane symbolu $:=$. V takom prípade sa výraz vypočíta s pôvodnou hodnotou x a potom sa vypočítaná hodnota priradí premennej x. Napríklad výraz

$$x := x + 1$$

znamená zväčšenie pôvodnej hodnoty x o jednotku, výraz

$$x := -x$$

zmenu znamienka x na opačné.

Pripúšťame i možnosť súčasného priradenia hodnôt viacerým premenným. Zápis

$$(x, y) := (e, f)$$

znamená

$$x := e \quad \text{a} \quad y := f$$

Zámenou hodnôt x a y môžeme napríklad zapísať nasledovne

$$(x, y) := (y, x)$$

4.2 Plánovanie rozkladom

Plánovanie rozkladom spočíva v rozložení základného problému na niekoľko podproblémov, z postupov riešení ktorých vieme spätne vytvoriť postup riešenia pôvodného problému.

Rozklad pôvodného problému na podproblémy môže byť taký, že postup riešenia pôvodného problému možno v etape realizácie postupu získať na základe postupu riešenia jediného z podproblémov, na ktoré sme pôvodný problém rozložili. Takémuto rozkladu budeme hovoriť *disjunktívny rozklad.* Ak musíme nájsť a aj realizovať postupy riešení všetkých podproblémov, na ktoré sme rozložili pôvodný problém, hovoríme o *konjunktívnom rozklade.* V prípade, keď postup riešenia pôvodného problému získame niekoľkonásobným opakovaním postupu riešenia niektorého z problémov, na ktoré sme pôvodný problém rozložili, hovoríme o *repetičnom rozklade.*

4.2.1 Disjunktívny rozklad

O *disjunktívnom rozklade* hovoríme vtedy, keď pôvodný problém rozkladáme na podproblémy, spomedzi postupov riešení ktorých budeme v etape realizácie postupu riešenia pôvodného problému realizovať iba jediný.

Rozoznávame dva druhy disjunktívneho rozkladu:

— *podmienený,* keď vieme explicitne stanoviť podmienku, určujúcu, postup riešenia ktorého podproblému je potrebné kedy realizovať,

— *nepodmienený,* ak takúto podmienku nepoznáme.

Podmienený disjunktívny rozklad používame napríklad pri riešení kvadratickej rovnice. Vzorec, ktorý treba použiť na výpočet koreňov, závisí totiž od hodnoty diskriminantu.

V programoch vedie použitie tohoto druhu rozkladu ku konštrukcii tvaru, podobného nasledujúcemu:

Konštrukcia I.

rob ak *podmienka:* krok **inak**
  ak *podmienka:* krok **inak**
    ⋮
  ak *podmienka:* krok **inak**
koniec

Výrazy rob, koniec, ak, inak sú príkazmi hypotetického programovacieho jazyka, ktorý v tomto článku používame. V reálnych programovacích jazykoch majú všetky svoje ekvivalenty. Krok chápeme ako krok v postupe tvorby riešenia daného problému. Preto jeho špecifikácia vedie vždy k nutnosti tvorby postupu riešenia problémov až dovtedy, kým sa problém, ktorý je krokom predstavovaný, neidentifikuje ako triviálny.

Postup riešenia kvadratickej rovnice môžeme teraz zapísať takto:

rob ak $D \geqq 0$: $(x_1, x_2) := \left(\frac{-b+\sqrt{D}}{2a}, \frac{-b-\sqrt{D}}{2a}\right)$ **inak**

ak $D = 0$: $(x_1, x_2) := \left(\frac{-b}{2a}, \frac{-b}{2a}\right)$ **inak**

ak $D \leqq 0$: $(x_1, x_2) := \left(\frac{-b+i\sqrt{D}}{2a}, \frac{-b-i\sqrt{D}}{2a}\right)$ **inak**

koniec

Skutočnosť, že jednotlivé podmienky sa navzájom nevylučujú, nie je v rozpore s našimi požiadavkami. Ak $D = 0$, ľubovoľný z uvedených krokov vedie k riešeniu. Ak nie je splnená ani jedna z podmienok, znamená to, že sme rozklad nepreviedli správne.

Nepodmienený disjunktívny rozklad si vyžaduje preskúšať postupne všetky možnosti, ku ktorým priviedol, až kým nenájdeme prvý podproblém, ktorého postup riešenia sme schopní vytvoriť. Vyjadrenie pravidla tvorby nepodmieneným disjunktívnym rozkladom je v programoch nasledovné

rob krok **alebo**
krok **alebo**
⋮
krok **alebo**
koniec

4.2.2 Konjunktívny rozklad

O konjunktívnom rozklade hovoríme vtedy, keď pôvodný problém rozkladáme na niekoľko podproblémov, z ktorých každý je potrebné vyriešiť (a v procese realizácie postupu riešenia pôvodného problému

každý postup riešenia realizovať), ak chceme vytvoriť (a realizovať) postup riešenia pôvodného problému. Podľa charakteru podproblémov, na ktoré sme pôvodný problém rozložili, rozoznávame dva druhy konjunktívneho rozkladu. Môže sa stať, že k nájdeniu postupu riešenia podproblému potrebujeme poznať riešenia niekoľkých predchádzajúcich podproblémov. Týmto vzniká na množine podproblémov usporiadanie, na základe ktorého môžeme všetky podproblémy usporiadať do postupnosti. V takýchto prípadoch hovoríme o *postupnostnom* konjunktívnom rozklade. Ak nutnosť vytvorenia postupnostného rozkladu nevzniká, hovoríme o *množinovom* konjunktívnom rozklade.

Známou metódou, ktorá v princípe vychádza z postupnostného rozkladu, je *matematická indukcia*. Problém pri dôkaze matematickou indukciou rozkladáme na dva podproblémy. Prvý podproblém spočíva v dôkaze existencie prvku s vyšetrovanou vlastnosťou P. Druhý podproblém spočíva v dôkaze možnosti prisúdiť vlastnosť R i ostatným prvkom množiny objektov, o ktorých treba dokázať, že majú vlastnosť P.

Dôležitú úlohu pri postupnostnom rozklade majú predpoklady, ktoré musia byť splnené, aby sme mohli vykonať nasledujúci krok riešenia. Napríklad pri matematickej indukcii je opodstatnené hľadať postup riešenia druhého podproblému iba vtedy, keď je splnený predpoklad, vyjadrený formulou

$$\exists x\ P(x)$$

V našom hypotetickom jazyku je postupnostný rozklad vyjadrený takouto konštrukciou:

Konštrukcia II.

rob krok
ak *predpoklad:* krok **pokračuj inak skonči**
ak *predpoklad:* krok **pokračuj inak skonči**
⋮
ak *predpoklad:* krok **pokračuj inak skonči**
ak *predpoklad:* krok
koniec

Ilustratívnym príkladom použitia konjunktívneho množinového rozkladu je rozklad dôkazu ekvivalencie na dve časti — nutnú a postačujúcu

podmienku. Je totiž ľahostajné, ktorú z týchto podmienok dokážeme ako prvú a ktorú ako druhú, musíme však dokázať obe.

4.2.3 Repetičný rozklad

O *repetičnom rozklade* hovoríme vtedy, keď pôvodný problém rozložíme na podproblémy s rovnakým postupom riešenia, takže postup riešenia pôvodného problému je vlastne niekoľkonásobným opakovaním postupu riešenia podproblémov. Môže sa stať, že súčasťou postupu riešenia podproblému je vykonanie celého toho istého postupu riešenia — že postup riešenia je udaný využívajúc sám seba ako svoju časť. Takémuto prípadu repetičného rozkladu hovoríme *rekurzívny*, predchádzajúcemu *iteračný* (alebo niekedy *cyklus*).

Iteratívny repetičný rozklad predstavuje v programoch výraz, podobný nasledovnému:

Konštrukcia III.

pokiaľ *podmienka:* krok

Rekurzívny repetičný rozklad má tvar:

záhlavie postupu ≡ krok

Poznamenávame, že nie všetky reálne programovacie jazyky dovoľujú takúto konštrukciu.

V prípade iteratívneho vytvárania postupu riešenia problémov sa postup, predpísaný v časti „krok" realizuje dovtedy, kým nie je splnená podmienka. Ak sa stane, že podmienka nie je splnená už pri prvom testovaní, postup, určený krokom, sa nevykoná ani raz.

Iteratívnym postupom môžeme (po stanovení začiatočných podmienok (ktorých?)) vypočítať skalárny súčin dvoch vektorov napríklad takto

pokiaľ $i \leqq n$: (SÚČIN, i) := (SÚČIN + $x_i y_i$, $i + 1$)

V prípade rekurzívneho repetičného rozkladu záhlavie postupu udáva názov postupu a jeho premenných. Tento názov spolu s premennými (obyčajne s inými hodnotami než aké sú v záhlaví postupu) sa nachádza i niekde napravo od spojky ≡, v časti krok. Tu predstavuje príkaz na opätovnú realizáciu postupu (s inými hodnotami premenných). Rekurzív-

ne môžeme vypočítať napríklad hodnotu faktoriálu čísla alebo hodnotu kombinačných čísel nasledujúcim spôsobom

$$n! \equiv \textbf{rob ak } n = 0 : 1 \textit{ inak ak } n \neq 0 : n \cdot (n-1)! \textbf{ koniec}$$

$$\binom{m}{n} \equiv \textbf{rob ak } n = 0 : 1 \textbf{ inak ak } m = n : 1 \textbf{ inak ak } (n \neq 0) \wedge (m \neq n) :$$
$$\binom{m-1}{n} + \binom{m-1}{n-1} \textbf{ koniec}$$

Dúfame, že práve uvedené príklady dostatočne jasne vysvetľujú i význam spojky $\equiv$.

4.3 Plánovanie preformulovaním

Preformulovať problém P znamená nahradiť ho iným problémom P', ktorého riešenie R' súvisí určitým spôsobom s riešením R pôvodného problému P. Podľa toho, aký je súvis medzi riešeniami R a R', rozoznávame tri spôsoby preformulovania (porovnaj napr. s [11]).

O *zjednodušujúcom* preformulovaní hovoríme v tom prípade, keď riešenie R' problému P' je špeciálnym prípadom riešenia R pôvodného problému P, alebo je R' iba určitou aproximáciou riešenia R. V matematike sa s typickým zjednodušujúcim preformulovaním stretávame pri aproximáciách funkcií: na základe nameraných hodnôt neznámej funkcie (R) v niekoľkých bodoch jej definičného oboru. Stanovujeme novú funkciu (R'), ktorá v istom zmysle aproximuje neznámu funkciu a v bodoch, v ktorých hodnotu R poznáme, sú hodnoty oboch funkcií rovnaké.

Ak je riešenie R problému P zhodné s riešením R' problému P', hovoríme o *ekvivalentnom* preformulovaní problému P na problém P'. K tomuto druhu preformulovania siahame napríklad pri riešení lineárnych rovníc. Jednotlivým krokom preformulovávania pôvodnej rovnice v nové hovoríme *ekvivalentné úpravy*.

Tretím spôsobom preformulovania je preformulovanie *zovšeobecnením*. Pri tomto spôsobe preformulovania je riešenie R pôvodného problému P špeciálnym prípadom riešenia R' nového problému P'. Napríklad ak (v reálnom obore) chceme vyriešiť rovnicu $x^2 + 3x - 10 = 0$ (P) vyriešime kvadratickú rovnicu $ax^2 + bx + c = 0 (P')$. Potom reálne korene (R)

pôvodnej rovnice (P) dostaneme ako špeciálny prípad koreňov (R') rovnice v uvedenom všeobecnom tvare.

Všetky spôsoby preformulovania vedú vždy k nutnosti riešiť nové problémy — tie, na ktoré sme pôvodný problém preformulovali.

Preformulovanie problémov je posledný spôsob vytvárania plánov postupov riešenia problémov, ktorým sa v tomto článku zaoberáme. V nasledujúcej časti sa zameriame na procesy, ktoré nám umožňujú odstraňovať chyby z vytvorených postupov riešení.[1)]

[1)] Dokončenie článku a zoznam literatúry budú uverejnené v nasledujúcom zväzku. (Pozn. redakcie)