



Algoritmus a algoritmizácia úloh

Počítačové spracovanie informácií
I. FEŠRR

Základné pojmy

Algoritmus

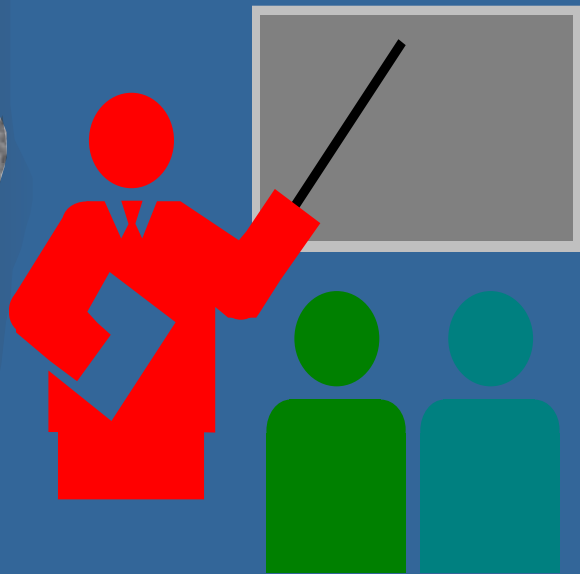
- presný a jednoznačný predpis postupu riešenia úlohy,
- proces transformácie vstupných údajov na výsledok,
- postupnosť elementárnych krokov (operácií), ktoré sa vykonávajú v určitom poradí



Algoritmom môže byť:

- návod na zostavenie nábytku z jednotlivých dielov,
- postup riešenia matematickej úlohy,
- recept na prípravu jedla,
- program zapísaný v ľubovoľnom programovacom jazyku...

Vlastnosti algoritmu



- **determinovanost'**
- **rezultatívnost'**
- **hromadnost'**

Determinovanosť

- algoritmus determinuje, t.j. presne určuje proces pretvárania vstupných údajov na výsledky,
- v každom kroku musí byť presne určené, ktorý krok sa vykoná ako ďalší,
- uskutočňovanie operácií nezávisí od vykonávateľa algoritmu.

Rezultatívnosť

- pre ľubovoľnú n -ticu vstupných údajov z určitej množiny M vedie algoritmus vždy po konečnom počte krokov k hľadanému výsledku,
- množinu M nazývame “oblasť použiteľnosti daného algoritmu”.

Hromadnosť

- algoritmus musí byť zostavený tak, aby riešil veľkú, obyčajne nekonečnú triedu úloh rovnakého typu,
- musí to byť popis riešenia nie jednej konkrétnej úlohy, ale celej skupiny príbuzných úloh, ktoré sa odlišujú len hodnotami vstupných údajov.

Ďalšie vlastnosti algoritmu

Okrem základných vlastností (determinovanosť, hromadnosť a rezultatívnosť) by algoritmus mal byť:

- elementárny,
- efektívny,
- čitateľný,
- modifikovateľný.

Elementárnosť algoritmu

- každý postup je možné realizovať rôznymi spôsobmi,
- pri návrhu algoritmu je potrebné dbať na to, aby jednotlivé kroky boli pre vykonávateľa algoritmu zrozumiteľné a jednoznačné.

Efektívnosť algoritmu

- určuje sa najmä vzhľadom na potrebný výpočtový čas a požadovanú kapacitu pamäti,
- efektívny algoritmus je taká postupnosť krokov, ktorá daný problém rieši s minimálnym počtom použitých prostriedkov v čo najkratšom čase.

Pri zložitých problémoch je efektívnosť algoritmu druhoradá - cieľom je zvyčajne vytvoriť akýkoľvek algoritmus, otestovať ho a až potom prípadne zvýšiť jeho efektívnosť.

Čitateľnosť a modifikovateľnosť algoritmu

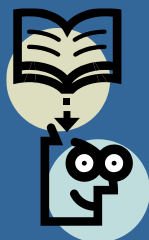
- je vhodné navrhovať algoritmus, ktorý je vhodne rozdelený na menšie, relatívne samostatné, logicky na seba nadväzujúce celky,
- **štruktúrovaný algoritmus** je ľahšie pochopiteľný a modifikovateľný.

Etapy algoritmizácie úloh

- **formulácia úlohy,**
 - jasná a jednoznačná formulácia a identifikácia,
 - stanovenie cieľa riešenia úlohy,
- **analýza úlohy,**
 - nájdanie algoritmu riešenia,
- **zostavenie riešiaceho algoritmu,**
 - syntetická etapa - popísanie logiky a postupu riešenia,
 - výsledkom je riešiaci algoritmus

Algoritmizácia

- schopnosť aktívne vytvárať algoritmy určené pre *nemysliace zariadenia*,
- je nevyhnutnou súčasťou schopnosti programovať na počítači,



Na tvorbu efektívnych a správnych algoritmov nestačí len zvládnutie algoritmického či programovacieho jazyka.

Je potrebná znalosť problémového prostredia a skúsenosť s formulovaním algoritmov.

Programovanie

Program je algoritmus napísaný v programovacom jazyku

Programovanie je konštruktívna myšlienková, ale aj praktická činnosť, kedy vytvárame nové programové produkty realizovateľné na počítači.

👍 *Programovaním sa učíme myslieť, organizovať svoje myšlienky a dokázať ich realizáciou poveriť počítač...*

Vytvorenie programu

Činnosti:

- Algoritmizácia daného problému – určenie vstupných a výstupných podmienok,
 - Vytvorenie programu (programového produktu) a vhodnej programovej dokumentácie,
 - Zapísanie a odladenie programu priamo na počítači
- 💣* Pri tvorbe zložitejších problémov je nevyhnutná dôkladná algoritmická a systémová príprava, ktorá jediná môže minimalizovať čas strávený pri počítači a vyhnúť sa nekonečnému prepracovávaní zdanlivo už hotových častí.*

Zápis algoritmu

Algoritmom môže byť napr. výpočtový proces, popis geometrickej konštrukcie, právny predpis, návod na vykonanie určitej činnosti a pod., je možné ho vyjadriť rôznymi prostriedkami:

- *slovným popisom,*
- *matematickými symbolmi (rovnice, matice, vzorce a pod.),*
- *konštrukčným postupom,*
- *grafickými značkami.*

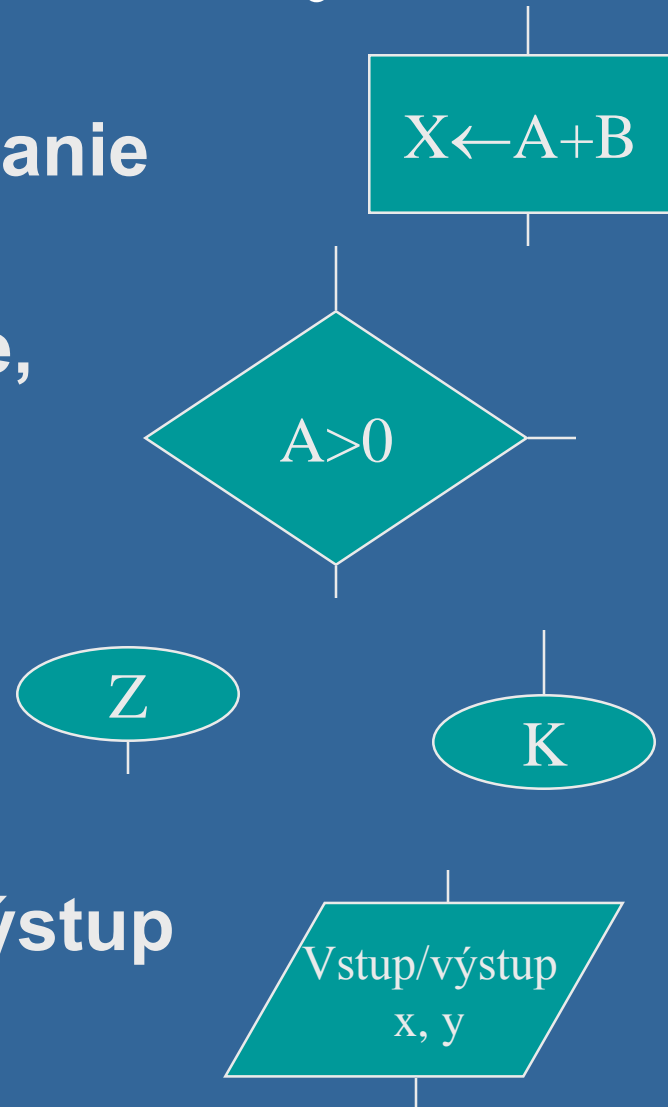
Vývojový diagram

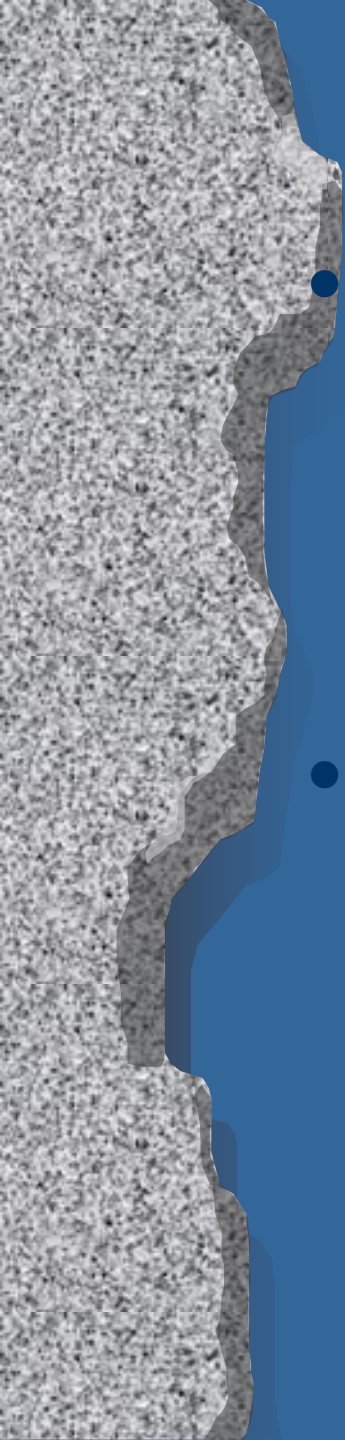
- grafická forma zápisu algoritmu,
- blokový diagram určitého procesu, ktorý vyjadruje jeho štruktúru a nadväznosť jednotlivých operácií,
- na kreslenie vývojových diagramov používame grafické značky.

Je definovaných niekoľko desiatok rôznych značiek, ktorých tvar je presne definovaný normou STN 36 9030 „Počítače a systémy spracovania údajov. Symboly vývojových diagramov pre systémy spracovania údajov“

Základné značky VD

- značka pre spracovanie
- značka pre vetvenie, rozhodovanie
- hraničná značka – začiatok, koniec
- značka pre vstup/výstup





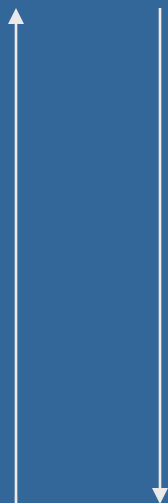
• spojky

• spojnice

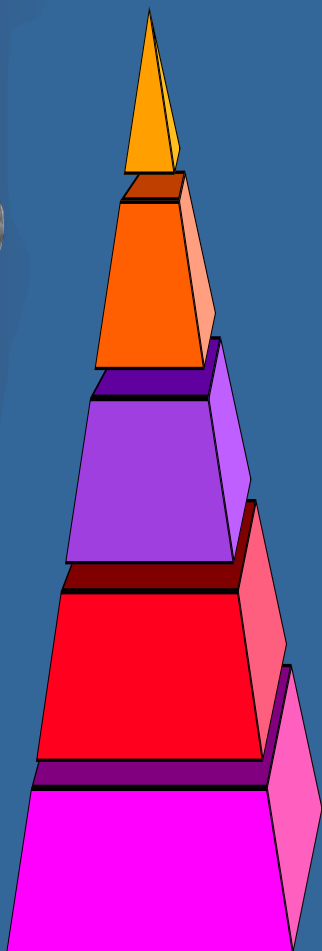
výstupná



vstupná



Druhy vývojových diagramov



Hrubé VD

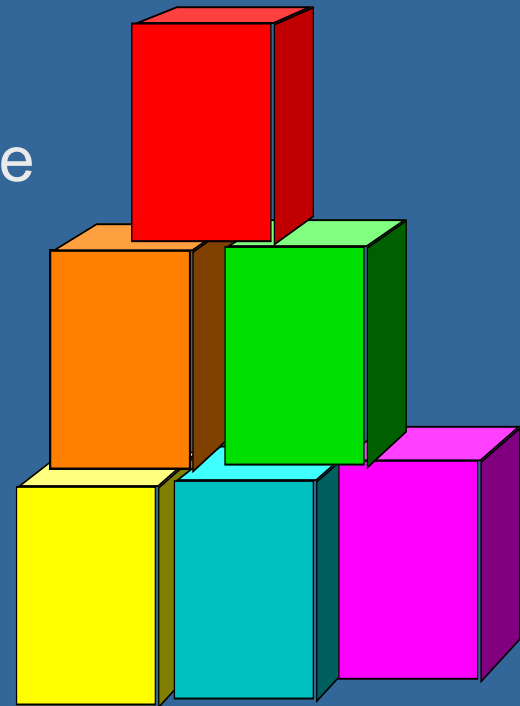
- riešenie rozsiahlych úloh, ktoré sú rozdelené na niekoľko čiastkových úloh (podprogramov)

Analytické VD

- podrobnejšie algoritmy riešenia VD programu
- detailne rozpracovaný postup riešenia - v každom bloku VD je spravidla jedna operácia

Vývojové diagramy programu

- VD priame
 - nedochádza k žiadnemu alternatívnemu postupu, proces je priamočiary (napr. sčítanie 4 čísel)
- VD s vetvením:
 - bez cyklu - bez opakovania (napr. maximum z 3 čísel)
 - s cyklom - s opakovaním (napr. súčet prvkov vektora, matice)



Príklad:

Vývojový diagram na výpočet súčtu väčšieho z čísel **a** alebo **b** a čísla **c**

Vstup:

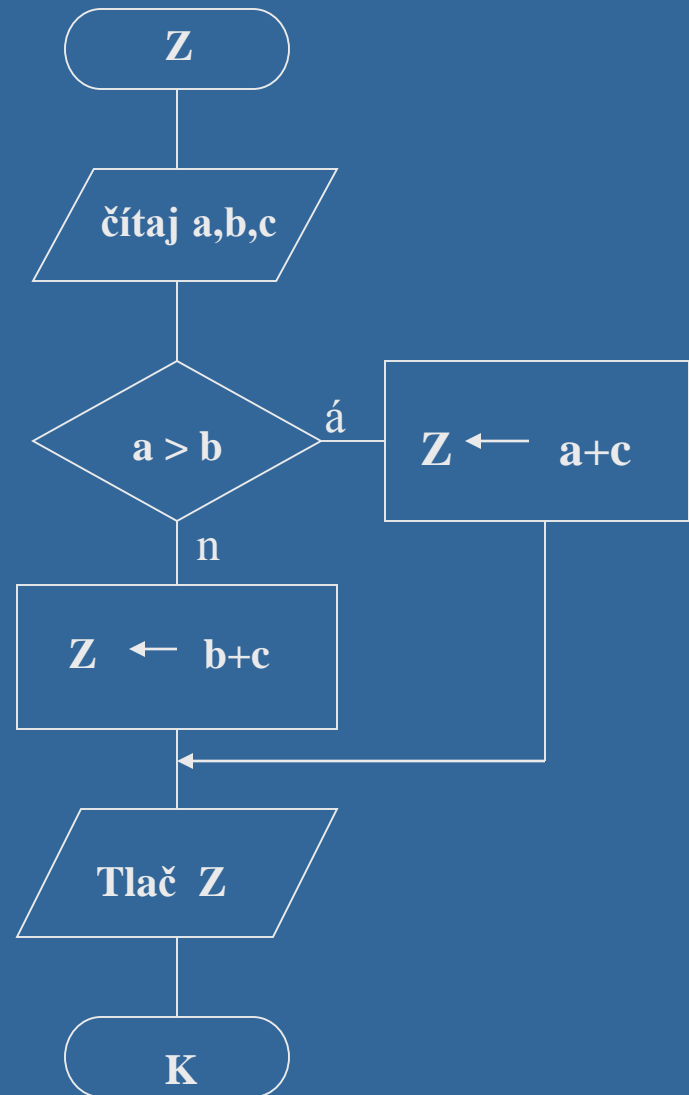
a, b, c – rôzne čísla

Výstup:

Z – výsledok – súčet

ak $a > b$ súčet bude $a+c$

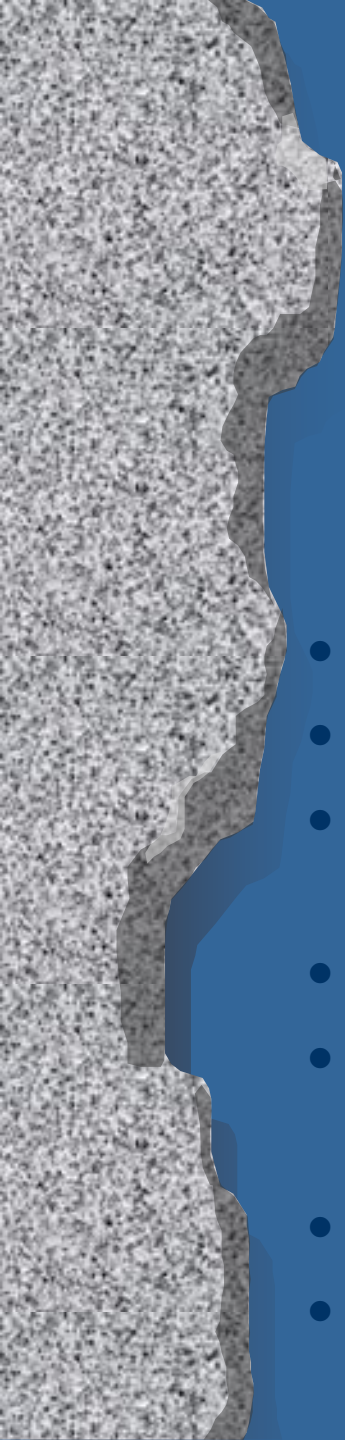
ak $b > a$ súčet bude $b+c$



Pochopenie pojmu *algoritmus* a procesu *algoritmizácie* si môžeme vysvetliť na nasledujúcom príklade:

- Prievozník má cez rieku previezť kozu, vlka a kapustu.
- K dispozícii má jeden čln, do ktorého sa okrem neho zmestí ešte jeden „pasažier“.

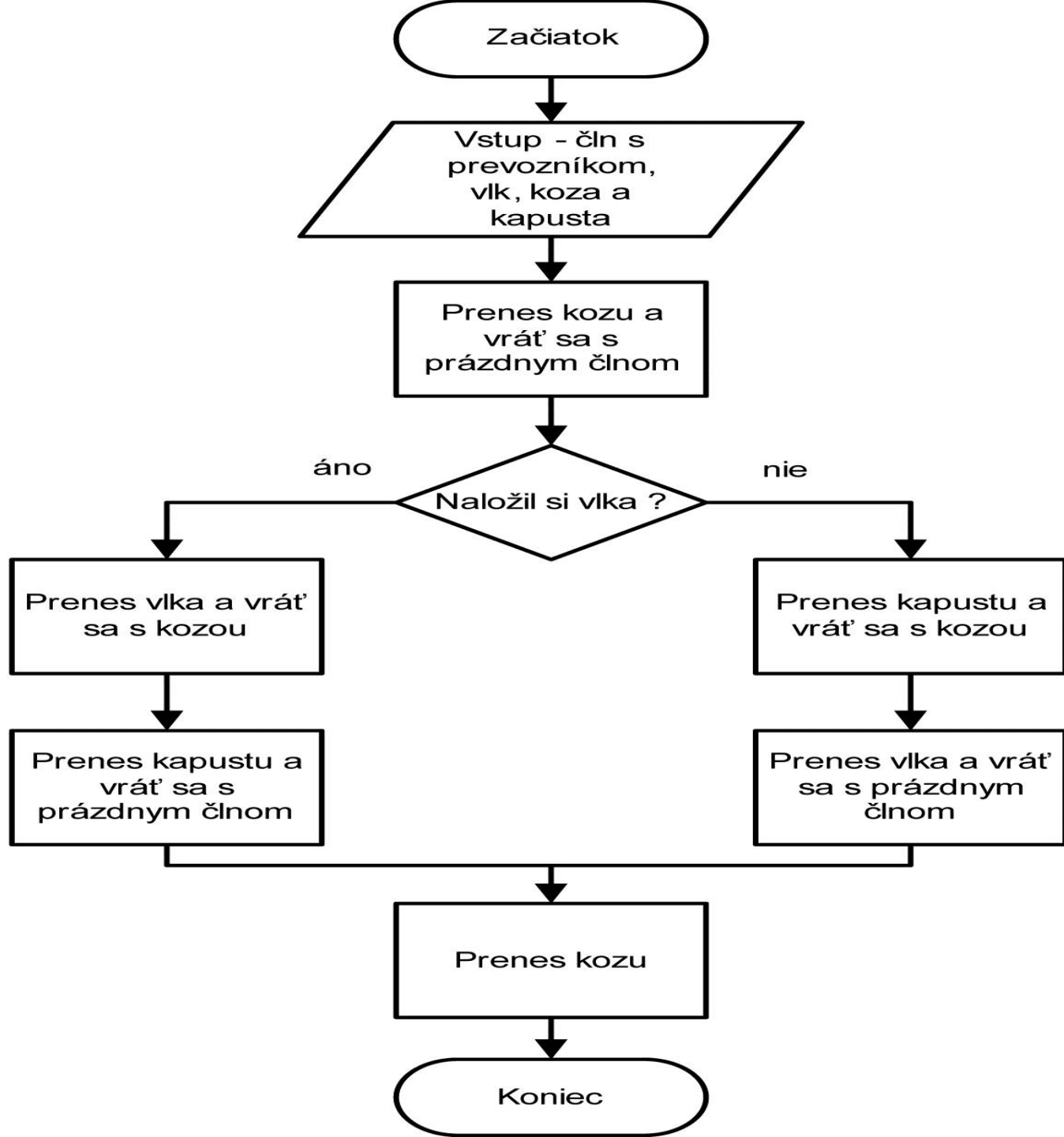
Úloha: Akým spôsobom splní úlohu, ak nesmie nechať na jednom brehu kozu s kapustou a vlka s kozou?



Pri riešení úlohy si treba uvedomiť, že prevozník môže prevážať „pasažiera“ nielen tam, ale aj naspäť a že vlk nie je vegetarián, preto môže zostať na brehu s kapustou.

Úlohu môžeme zapísať v nasledujúcich krokoch:

- **1. krok:** Prevozník prevezie na druhý breh kozu.
- **2. krok:** Vráti sa s prázdny člnom.
- **3. krok:** Na druhú stranu prevezie kapustu alebo vlka (má možnosť voľby).
- **4. krok:** Vráti sa s kozou, ktorú vyloží.
- **5. krok:** Prevezie vlka (ak v predchádzajúcom kroku previezol kapustu) alebo kapustu.
- **6. krok:** Vráti sa s prázdny člnom.
- **7. krok:** Prevezie kozu.



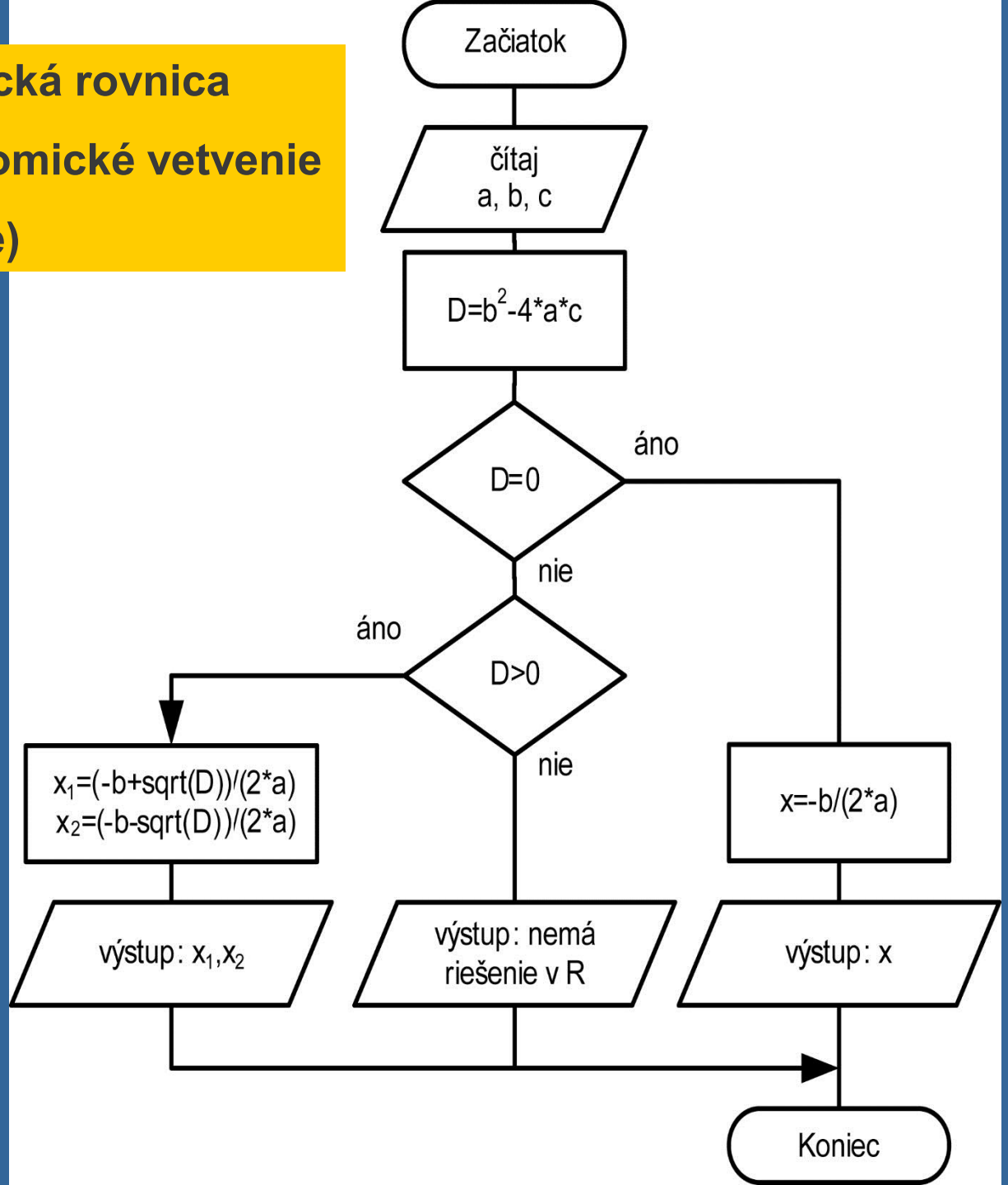
VD s vetvením bez opakovania

- znázorňujú proces, v ktorom nastáva **alternatívny postup** pozostávajúci minimálne z dvoch možností.

Príklady:

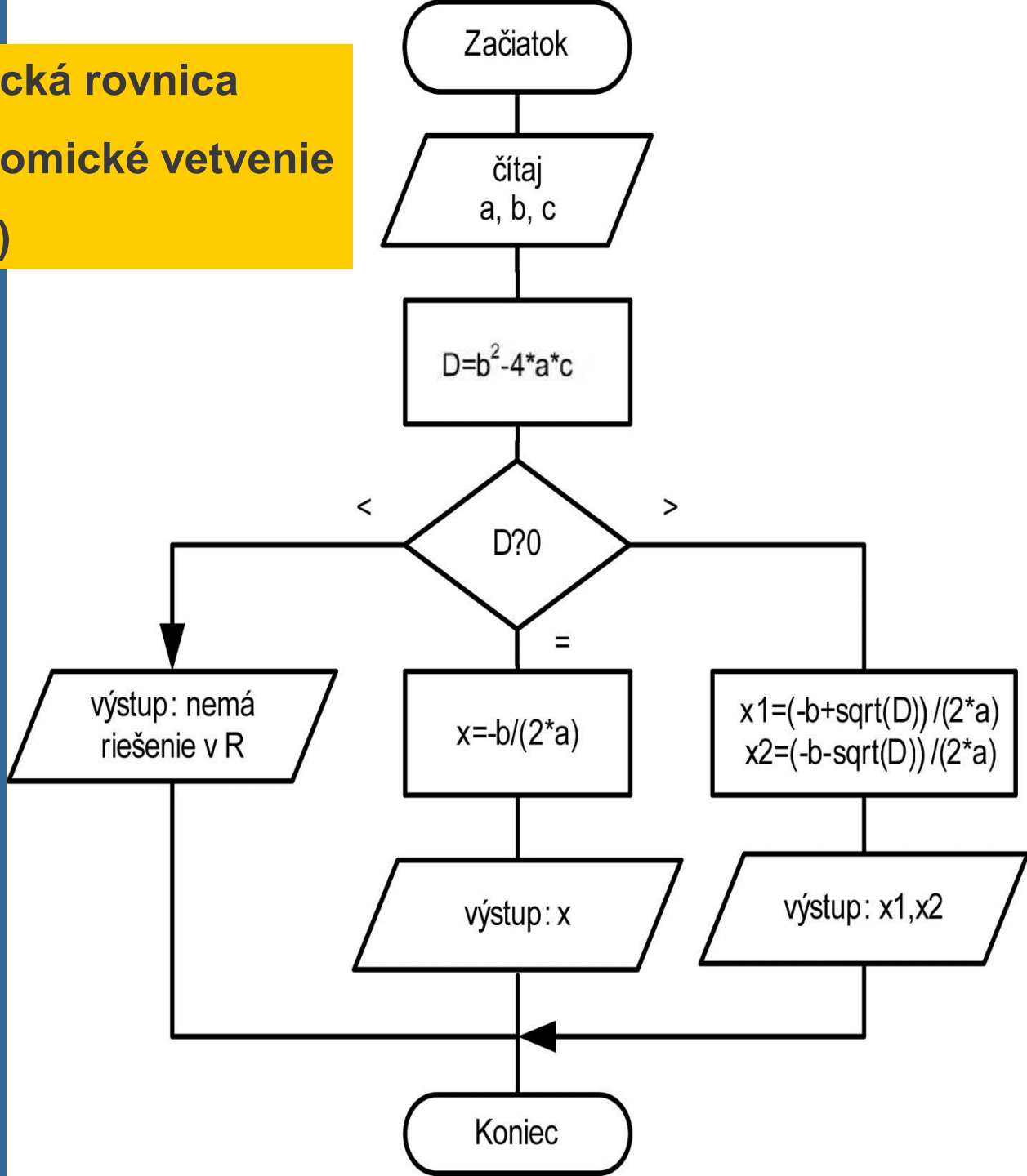
- *hľadanie najväčšieho, resp. najmenšieho čísla z 3 čísel,*
- *hľadanie 2 najväčších, resp. 2 najmenších čísel zo 4 čísel,*
- *výpočet koreňov kvadratickej rovnice...*

Kvadratická rovnica
- dichotomické vetvenie
(dvojité)



Kvadratická rovnica

- trichotomické vetvenie
(trojité)



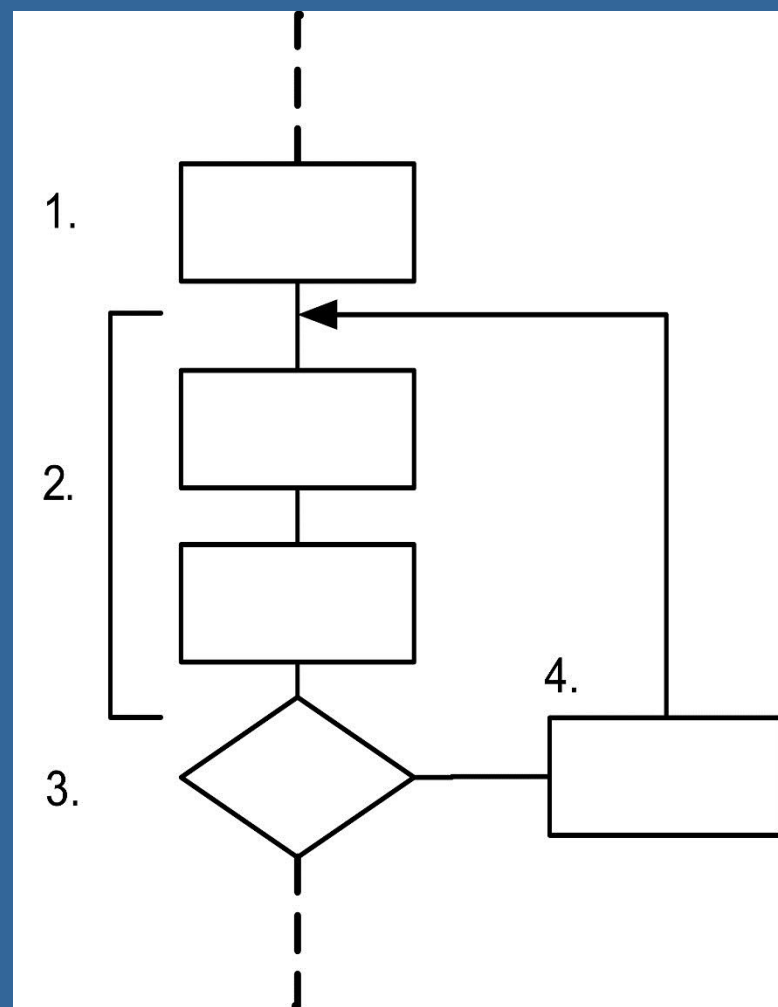
VD s vetvením s opakovaním

- jedna časť VD sa niekoľkokrát opakuje za **zmenených podmienok**,
- takáto časť VD sa nazýva **cyklus**,
- VD s vetvením s opakovaním = **VD s cyklom**

Cyklus vo VD

Časti cyklu vo VD:

1. prípravná časť cyklu,
2. operačná (vnútorná) časť cyklu,
3. koncová podmienka - rozhodovacia časť cyklu,
4. modifikačná časť cyklu.



Postup spracovania v cykle

- V **prípravnej časti** (1. fáza) sa priradí parametru cyklu počiatočná hodnota.
- **Operačná časť** cyklu (2. fáza) obsahuje operácie, ktoré sa majú opakovať, pričom počet opakovaní závisí od hodnoty parametra cyklu,
- **Koncová podmienka – rozhodovacia časť** cyklu (3. fáza)– hodnota parametra cyklu sa porovnáva s konečnou hodnotou parametra cyklu.
- **Modifikačná časť** (4. fáza) po vykonaní všetkých operácií v operačnej časti cyklu zmení hodnotu parametra pripočítaním konštantnej hodnoty, tzv. **kroku modifikácie**, k parametru cyklu a vráti riadenie procesu na začiatok 2. fázy.

Parameter cyklu

- hodnota, ktorá určuje počet opakovaní cyklu,
- môže ním byť buď jednoduchá alebo indexovaná premenná,
- parameter cyklu môže nadobúdať hodnoty v určitých hraniciach, vymedzených rozsahom riešenej úlohy.

Algoritmus sčítania n zložiek vektora (postupnosti)

$a\{a_i\}; i=1,2,3,\dots,n$

Jednotlivé kroky budú nasledujúce:

1. načítanie počtu spracovaných údajov - n ,
2. zadefinovanie a inicializácia výsledku sčítania - S ,
3. nastavenie parametra cyklu na počiatočnú hodnotu,
4. načítanie spracovávaného údaja - a_i ,
5. pripočítanie nového údaja k priebežnému súčtu,
6. testovanie konečnej hodnoty parametra cyklu,
7. ak parameter cyklu nenadobudol konečnú hodnotu, jeho modifikácia,
8. ak parameter cyklu nadobudol svoju konečnú hodnotu, ukončenie cyklu a výstup výsledku.

