Výrazy, příkazy, cykly

Výrazy

- je **předpis na získání hodnoty**.
- Vytváří se z **operandů** (čísla ,konstatnty, proměnné ...) , **operátorů** (určují druh operace , např. + , - , * , / , ..) a kulatých závorek.
- Zápis výrazů je lineární, není možné používat zlomky ani exponenty

  Např. Zápis diskriminantu: D:=B*B-4*A*C

  x1:=-B+SQRT(D)/(2*A)

Výrazy

Matematický zápis	Zápis v Pascalu
A.B+C	A*B+C
(A.K):B	(A*K)/B
A^2	sqr(A)
$\sin^2 x$	sqr(sin(x))
Xe(1,5)	(X<5) and (X>1)

Výrazy

- Operace se provádí v pořadí podle tzv. **priority operátorů**.
- Pro základní operátory vypadá takto:
  1. not
  2. * , / , div , mod , and
  3. + , - , or
  4. < , > , >= , <= , = , <> ... relační operátory
- Priorita logických operátorů (not,and,or) je vyšší než relačních operátorů, proto musíme požívat závorky - např. (X<5) and (X>1)

Příkazy

- Předepisují nějakou akci
- Po kompilaci se přeloží do instrukcí, které vykonají nějaké matematicko-logické operace s paměťovými místy, pracují se soubory, periferiemi apod.

Příkazy

jednoduché	strukturované
přiřazovací příkaz	složený příkaz
příkaz procedury	podmíněný příkazy (IF a CASE)
příkaz skoku	příkazy cyklu (FOR, REPEAT, WHILE)
prázdný příkaz	příkaz WITH

Přiřazovací příkaz
proměnná := výraz;

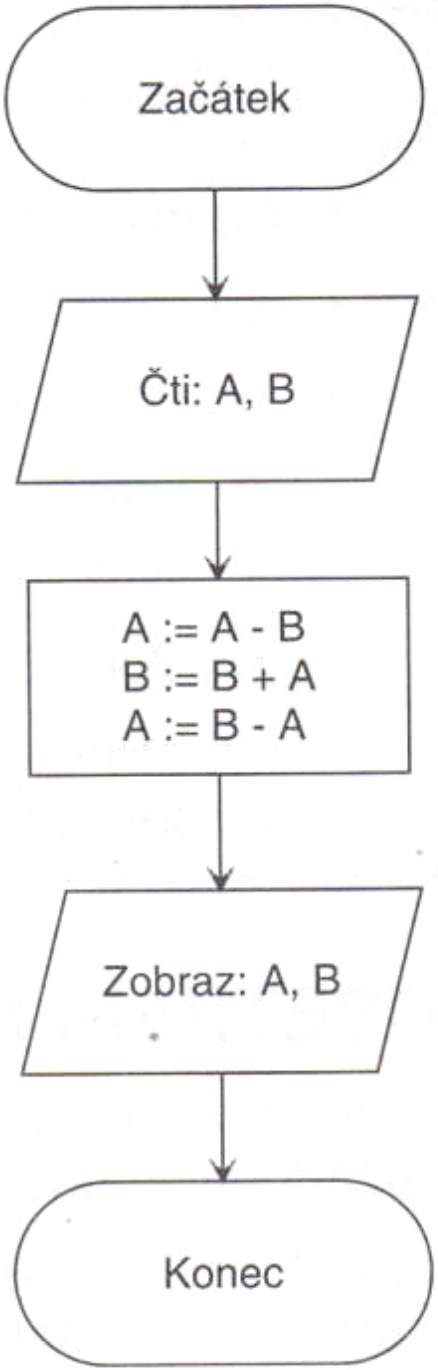
- Je to předpis k provedení akce:
  - nejprve se vyhodnotí výraz za symbolem :=
  - potom se jeho výsledná hodnota přiřadí (uloží) do proměnné před symbolem :=
- Výraz musí být **kompatibilní vzhledem k přiřazení**
  - protože proměnná může nabývat jen hodnoty typu podle své deklarace, musí hodnota výrazu patřit do množiny hodnot specifikované typem proměnné

Příkaz procedury

- slouží k aktivaci (vyvolání) procedur (podprogramů) standardních nebo programátorem definovaných
- Mezi standardní patří např. procedury pro vstup a výstup **write(parametry)** a **read(parametry)**
- Programátorem definované se musí nejprve deklarovat (poznáme později) a pak je aktivujeme názvem s parametry

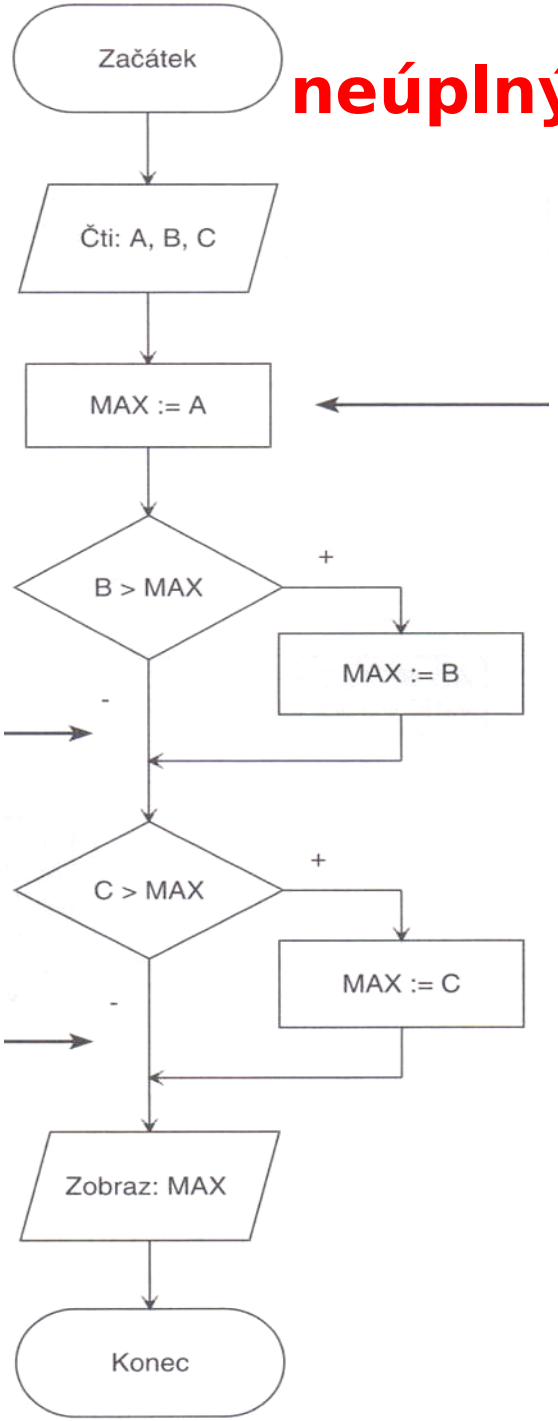
Složený příka

- Zapsání posloupnosti sekvenčních příkazů, které se vykonávají bezprostředně po sobě



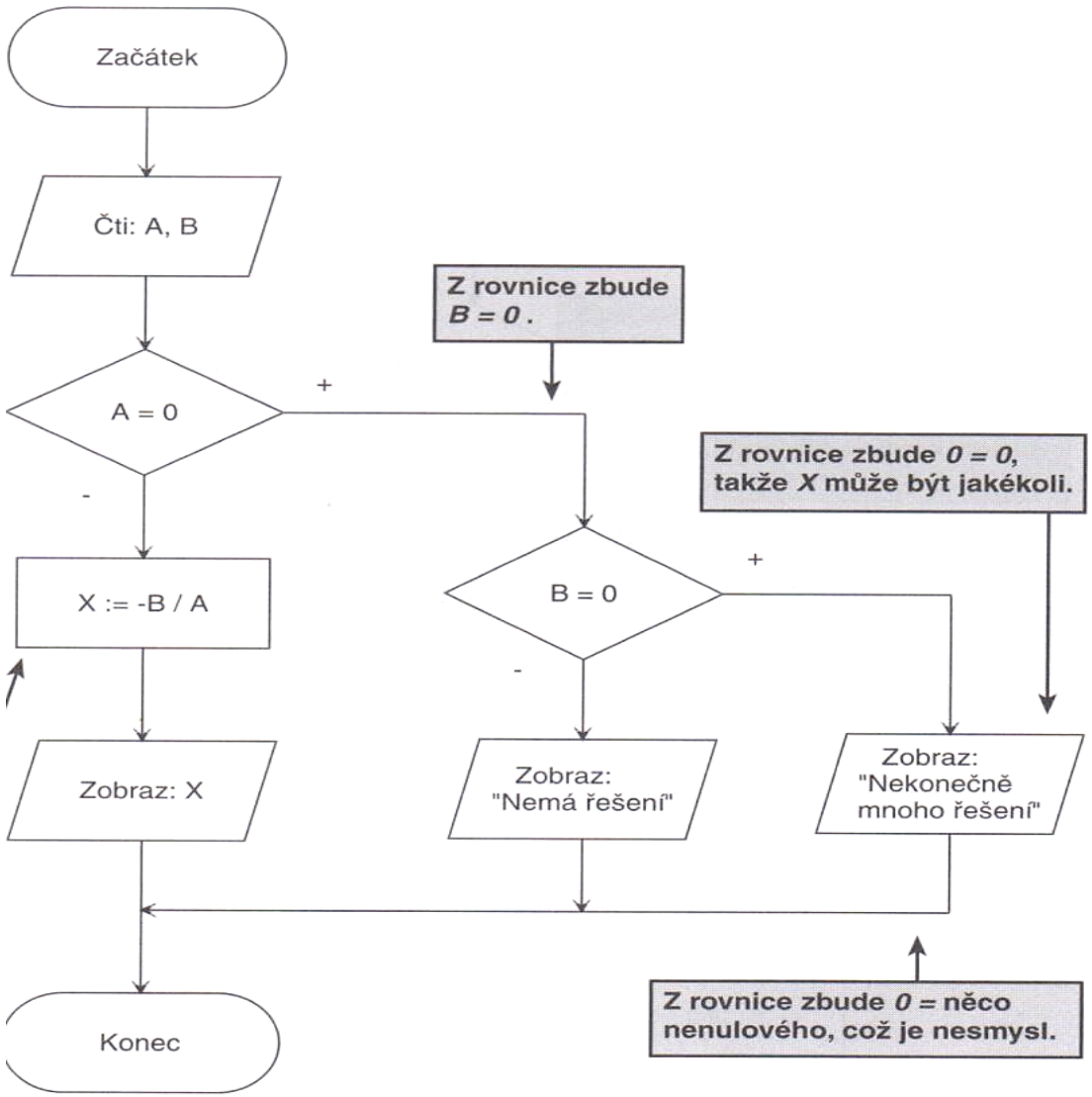
Podmíněné příkazy (IF a CASE)

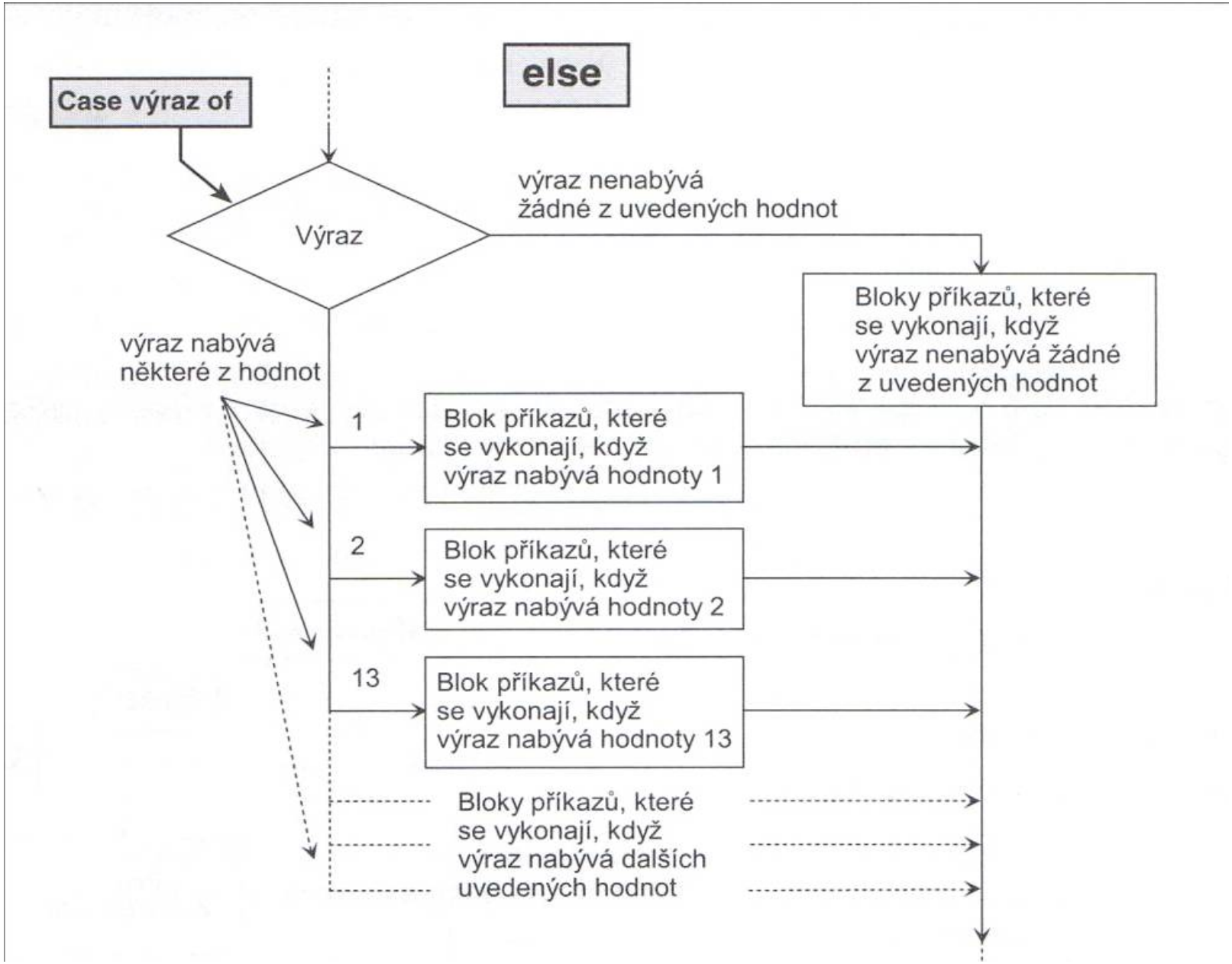
- **Příkaz If**
  Umožňuje vybrat k realizaci jeden ze dvou příkazů na základě splnění nebo nesplnění určité podmínky. Může být úplný nebo neúplný (jedna z akcí schází).
- **Příkaz case** - umožňuje provedení jednoho příkazu z několika alternativních



neúplný

úplný



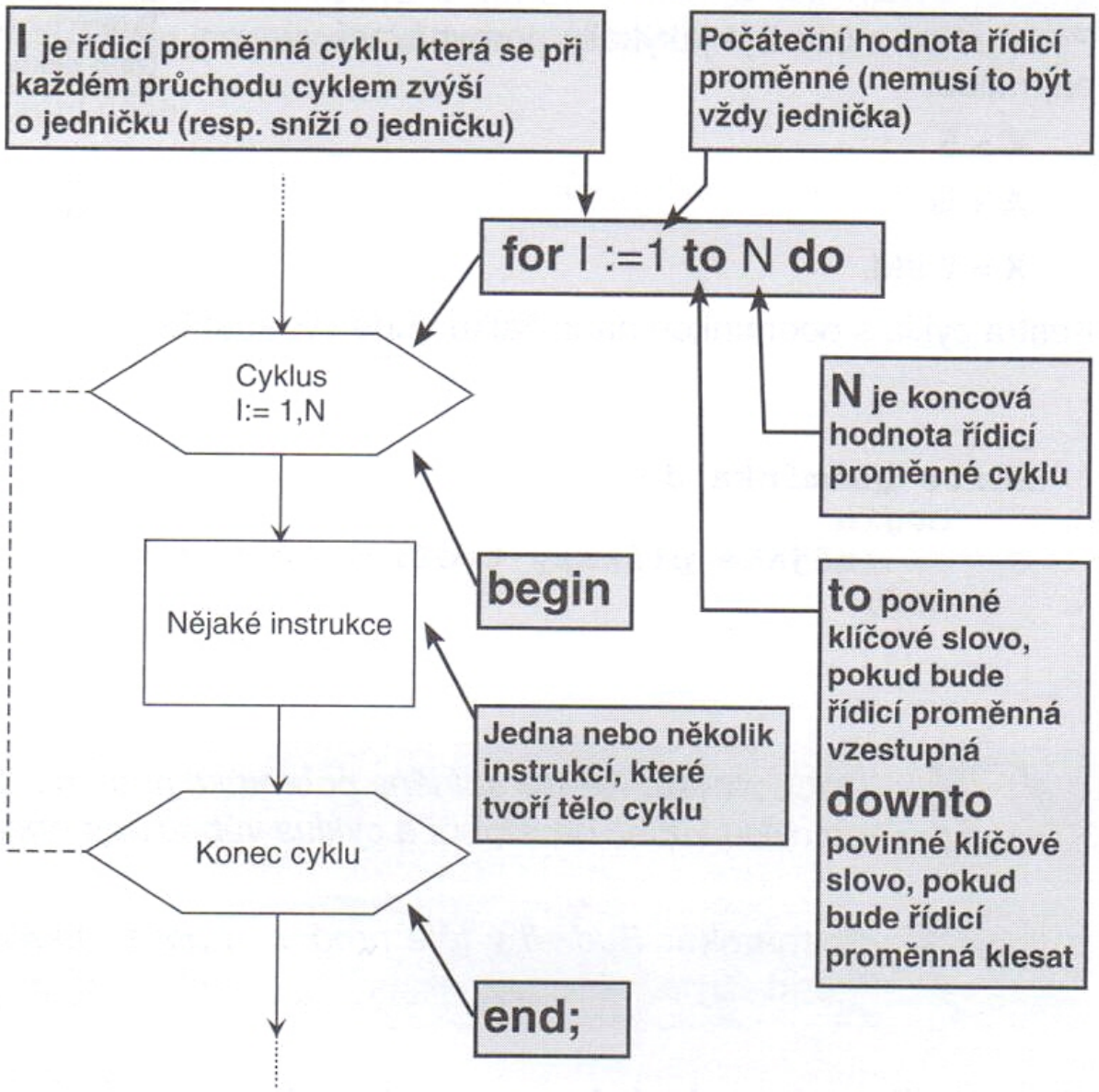

Case výraz of
else
Výraz
výraz nenabývá
žádné z uvedených hodnot
Bloky příkazů, které
se vykonají, když
výraz nenabývá žádné
z uvedených hodnot
výraz nabývá
některé z hodnot
1
Blok příkazů, které
se vykonají, když
výraz nabývá hodnoty 1
2
Blok příkazů, které
se vykonají, když
výraz nabývá hodnoty 2
13
Blok příkazů, které
se vykonají, když
výraz nabývá hodnoty 13
Bloky příkazů, které
se vykonají, když
výraz nabývá dalších
uvedených hodnot

CYKLY

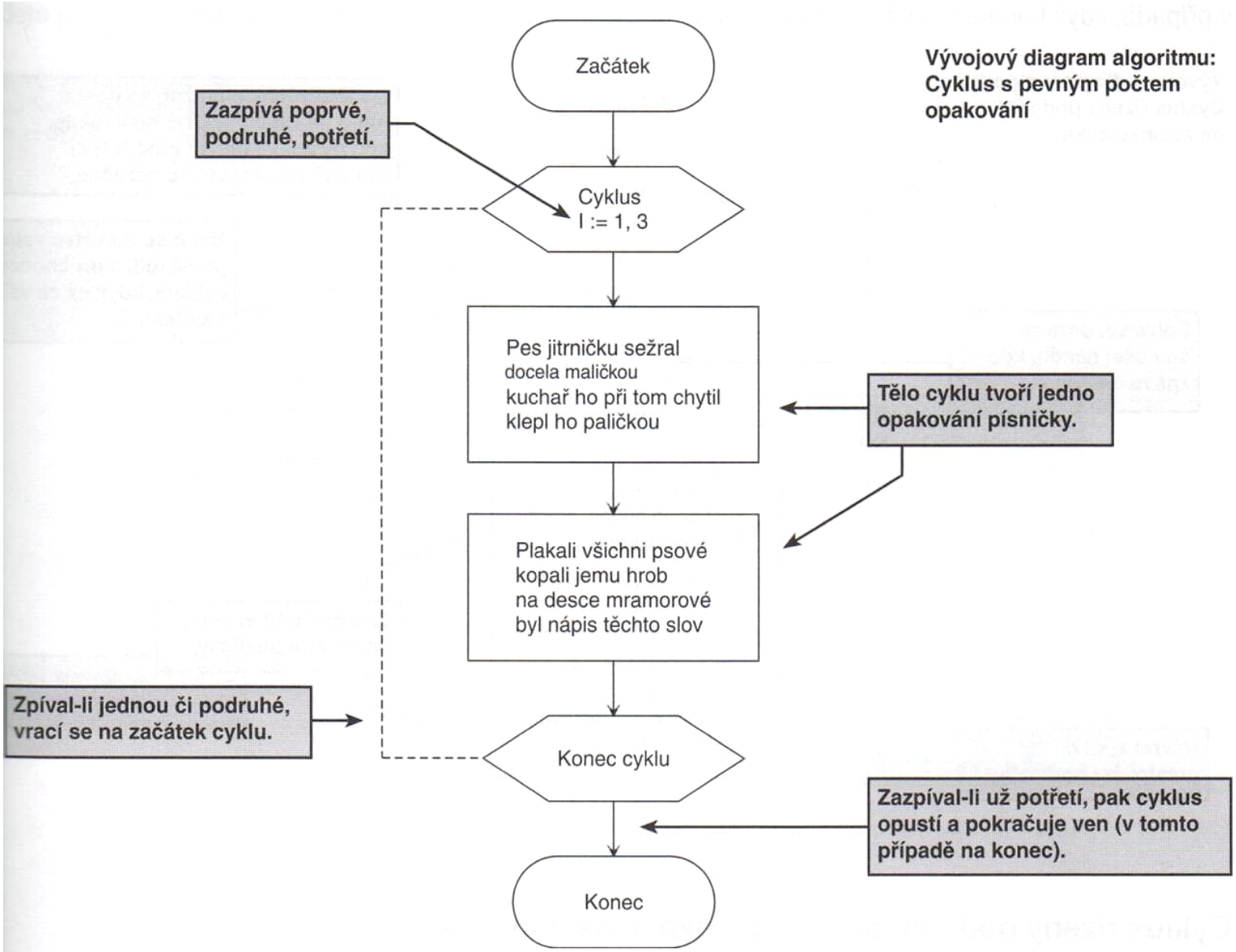
- Opakování stejné části algoritmu (příkazů nebo posloupnosti příkazů) vícekrát (se stejnými nebo pokaždé s jinými daty)
- Žádný cyklus nesmí běžet nekonečně dokola – nutno definovat za jakých podmínek se bude cyklus opakovat a kdy má skončit
- Rozeznáváme 3 typy cyklů

Cykly

S pevným počtem opakování



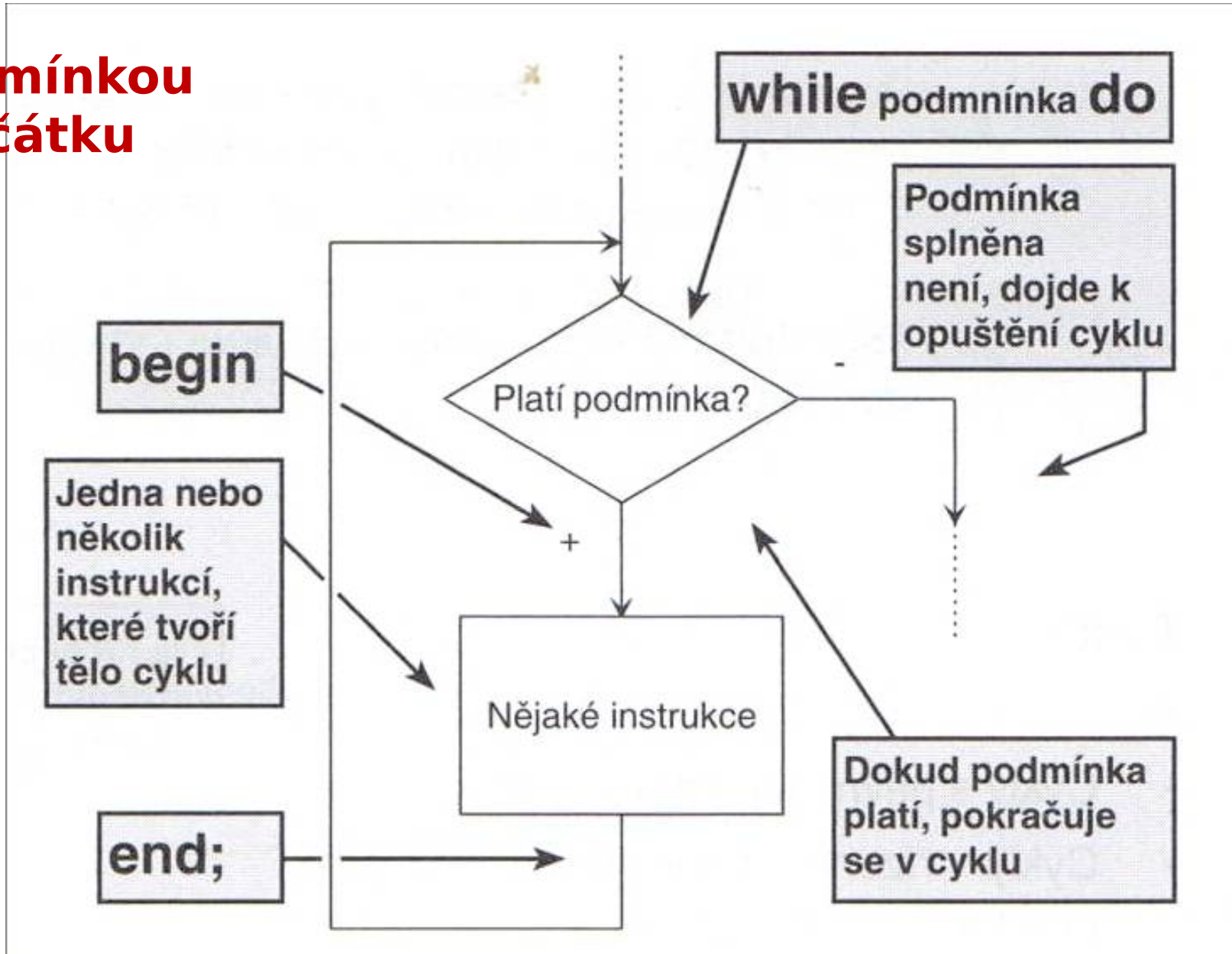
Cyklus s pevným počtem opakování



Vývojový diagram algoritmu: Cyklus s pevným počtem opakování

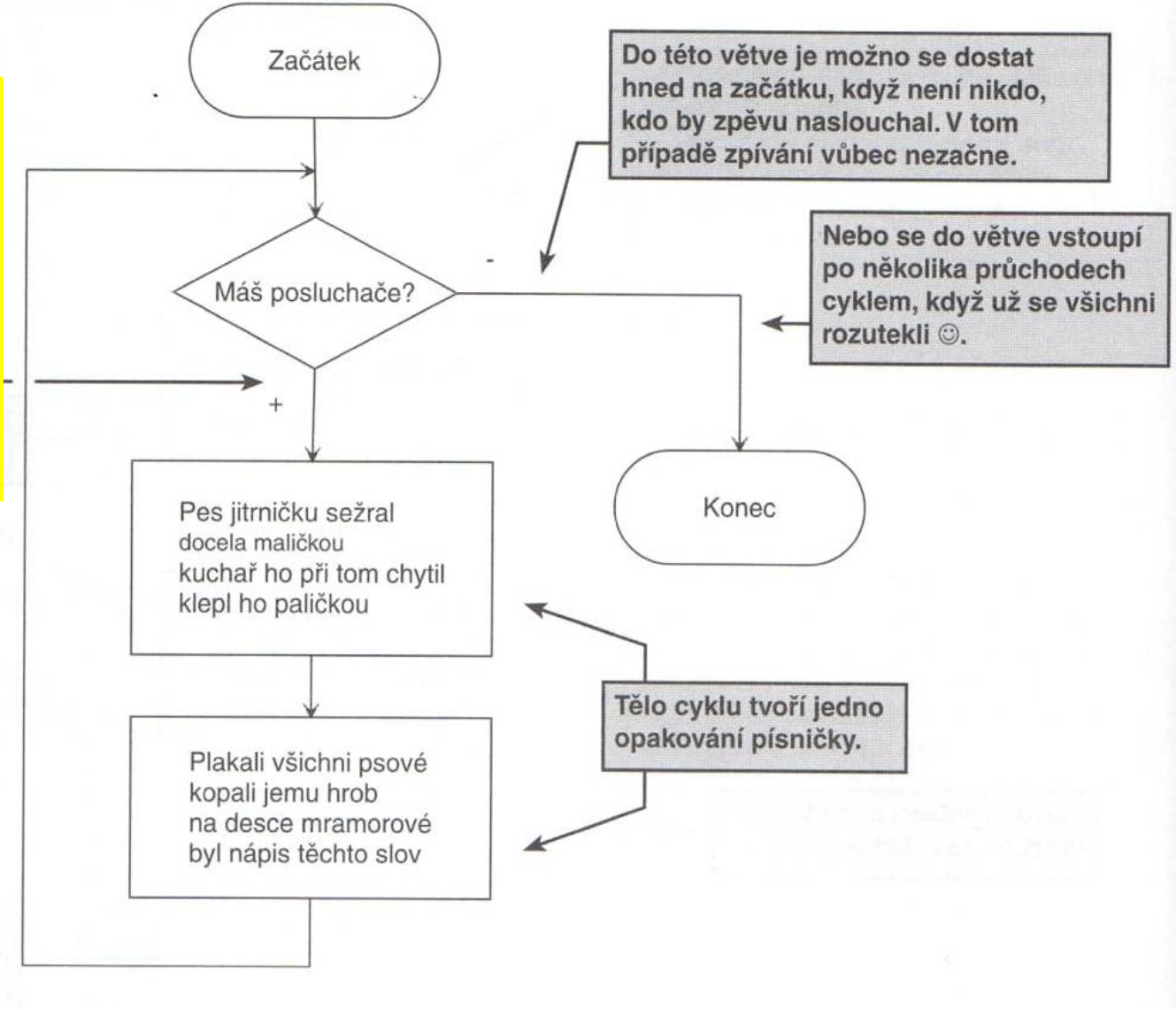
Cykly

S podmínkou na začátku



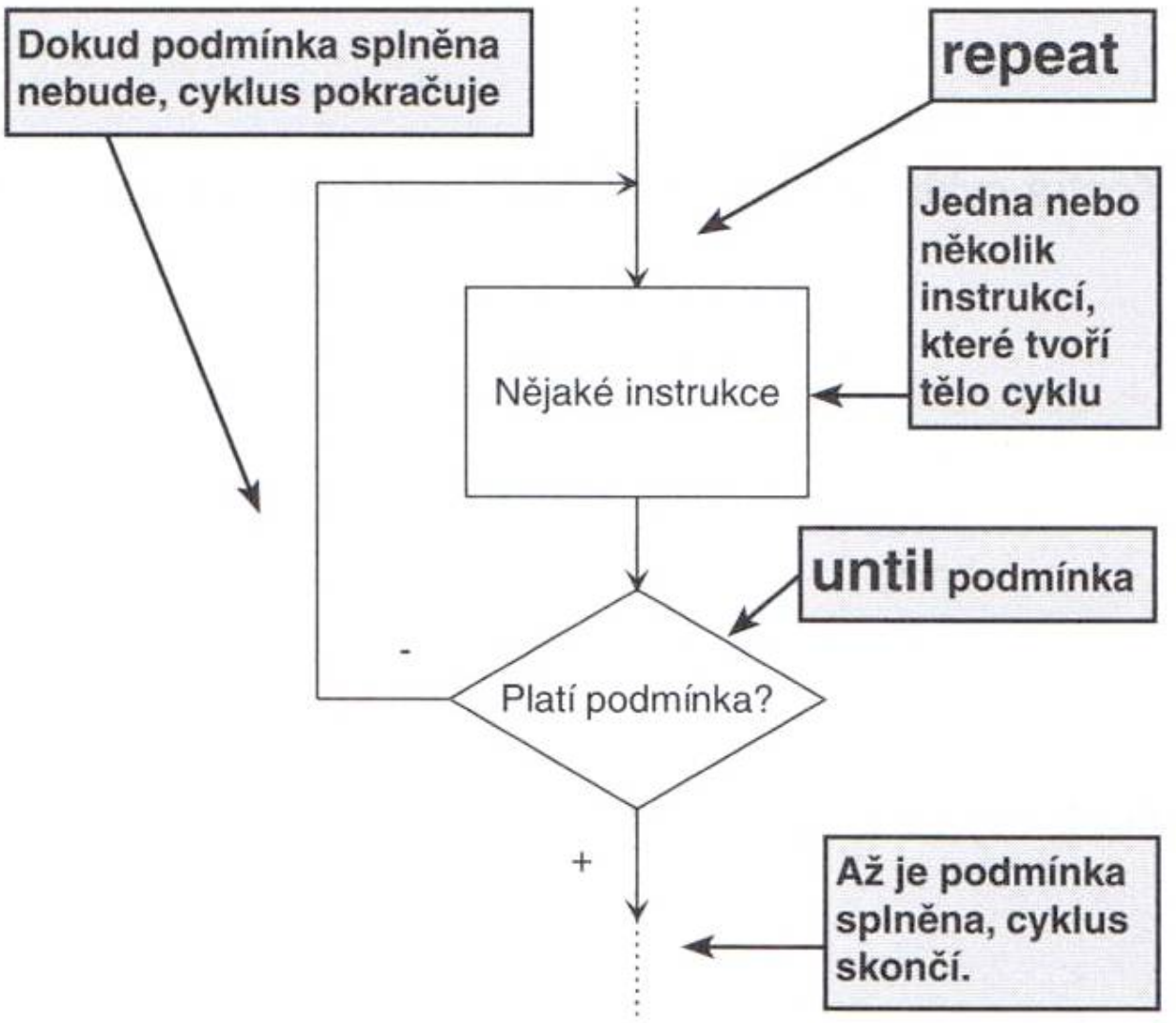
Cyklus s podmínkou na začátku

Není-li podmínka splněna hned na začátku – cyklus vůbec neproběhne



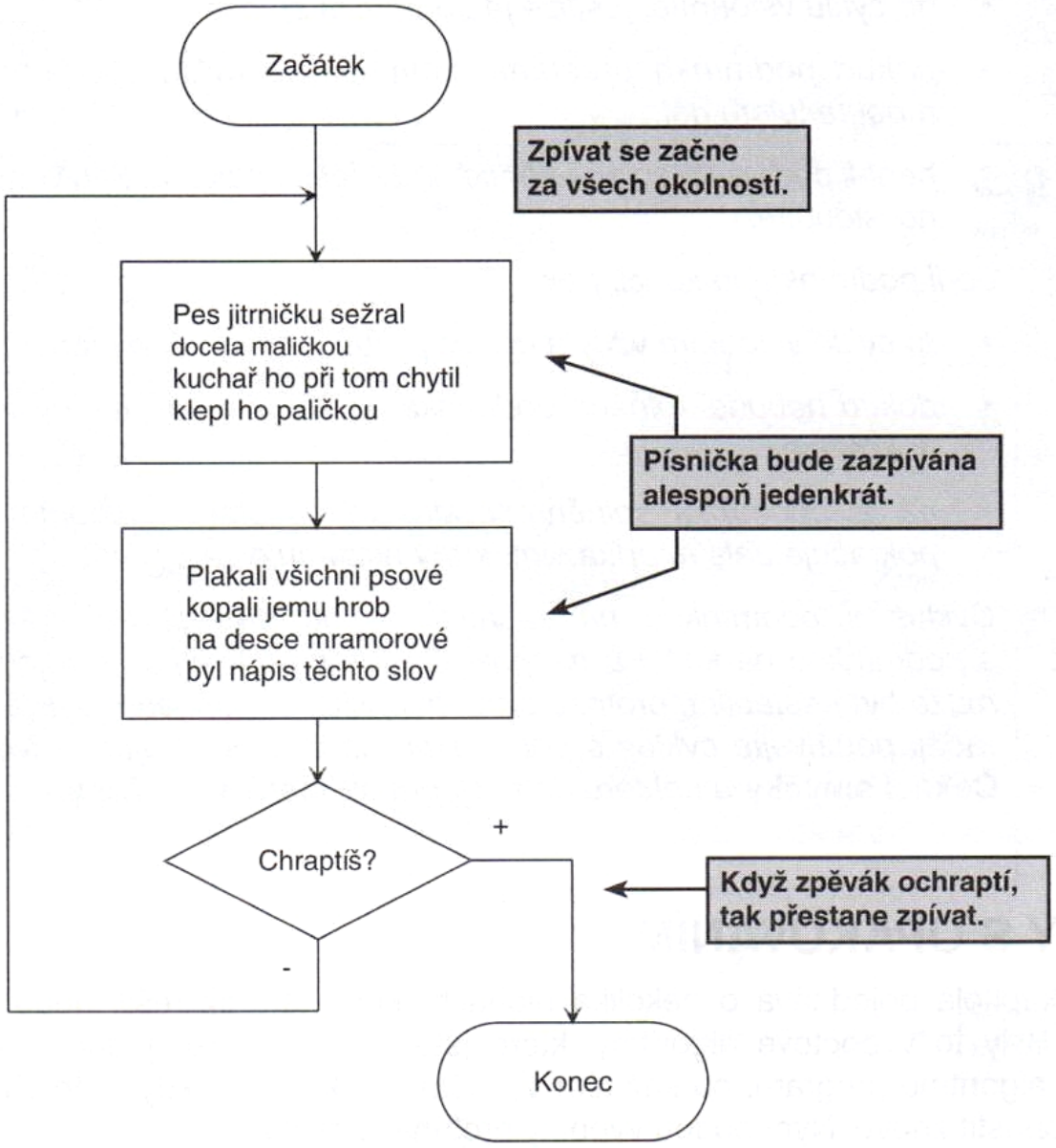
Cykly

S podmínkou na konci



Cyklus s podmínkou na konci

Cyklus vždy proběhne aspoň jednou, bude se tak dlouho opakovat, dokud nedojde ke splnění podmínky



Pro cykly řízené podmínkou platí, že:

Je-li podmínka na začátku, pak:

• *do cyklu vstoupíte, jestliže je podmínka splněna;*

• *pokud podmínka přestane platit, pak z cyklu vystoupíte*

a *pokračujete dál;*

• *není-li podmínka splněna hned na začátku, pak do cyklu vůbec*

nevstoupíte.

Je-ti podmínka na konci, pak*:*

• *do cyklu vstoupíte vždy a cyklus proběhne nejméně jednou;*

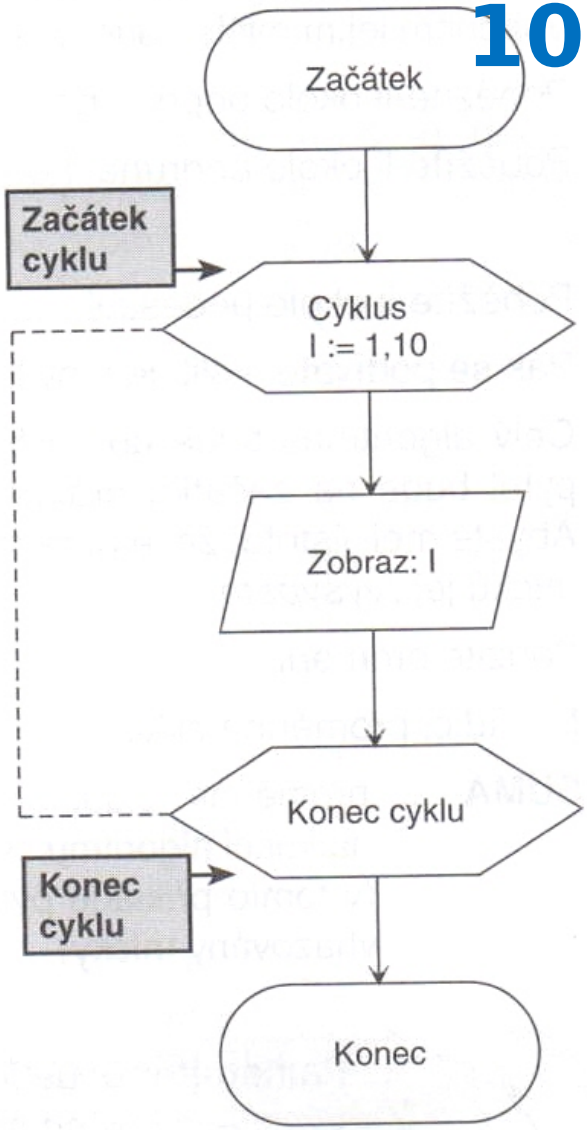
• *dokud nebude splněna podmínka na konci cyklu, tak cyklus*

pokračuje;

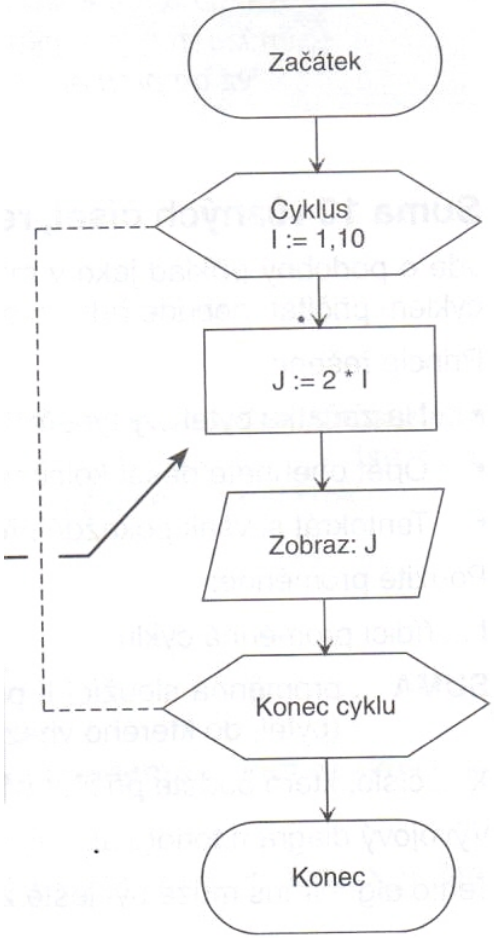
• až *je podmínka splněna, cyklus je opuštěn a algoritmus*

pokračuje dalším příkazem, který následuje za cyklem.

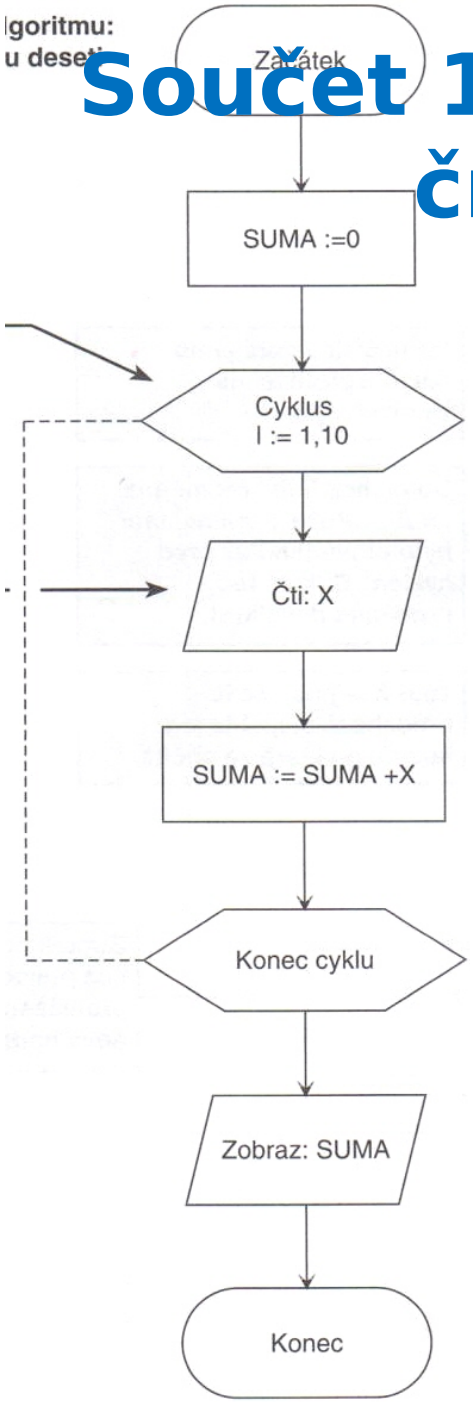
Zobrazení čísel od 1 do 10



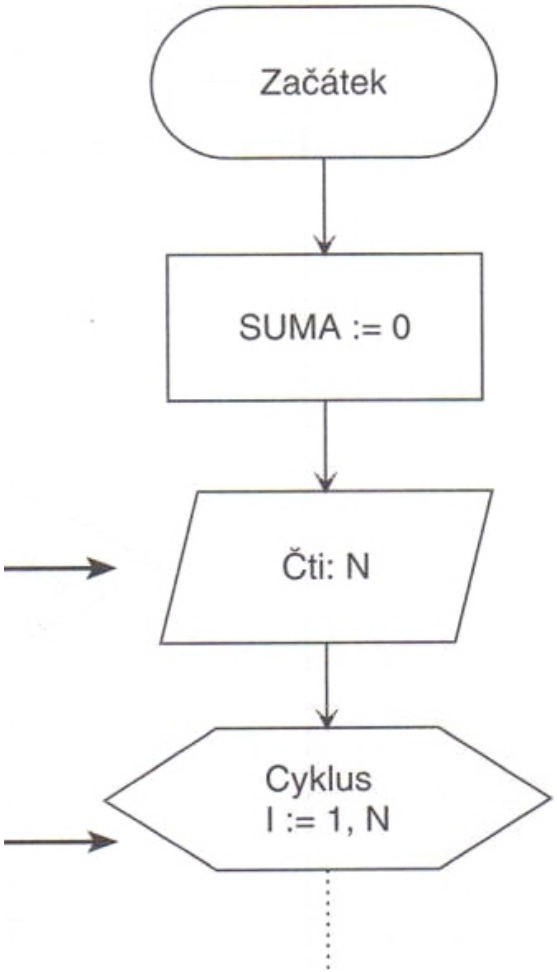
Zobrazení čísel od 2 do 20 jen sudé



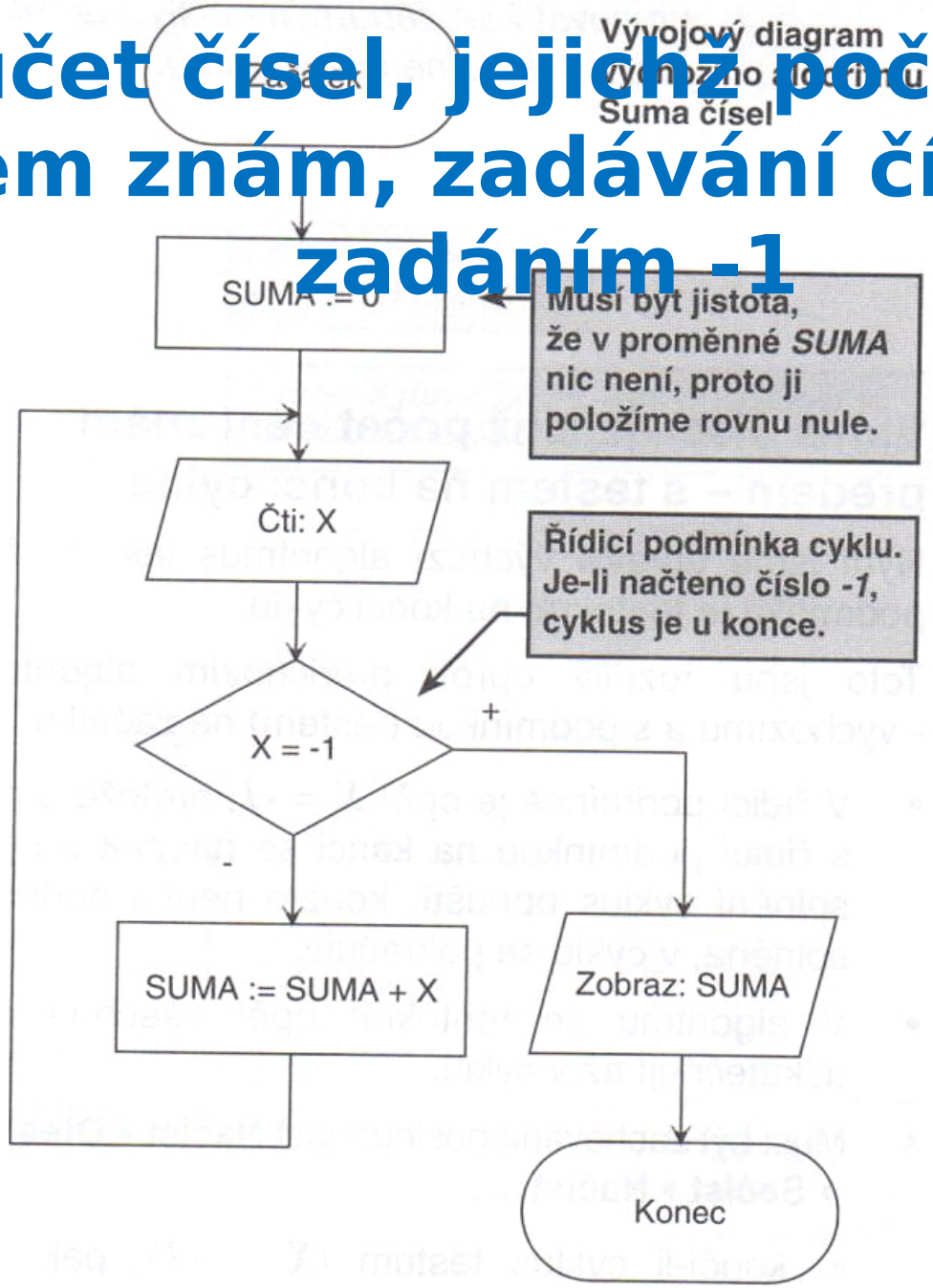
Součet 10 různých čísel



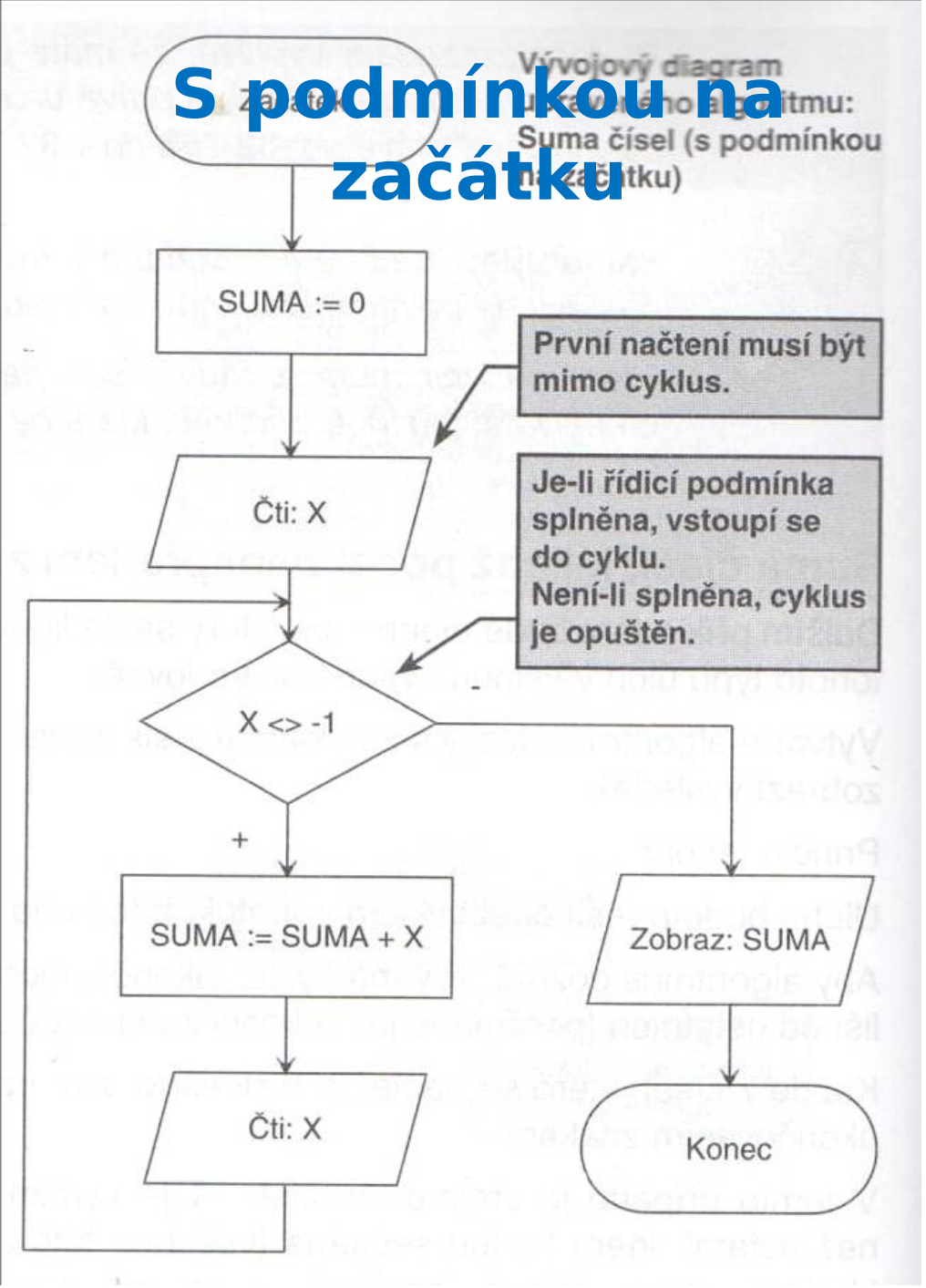
Součet *n* různých čísel



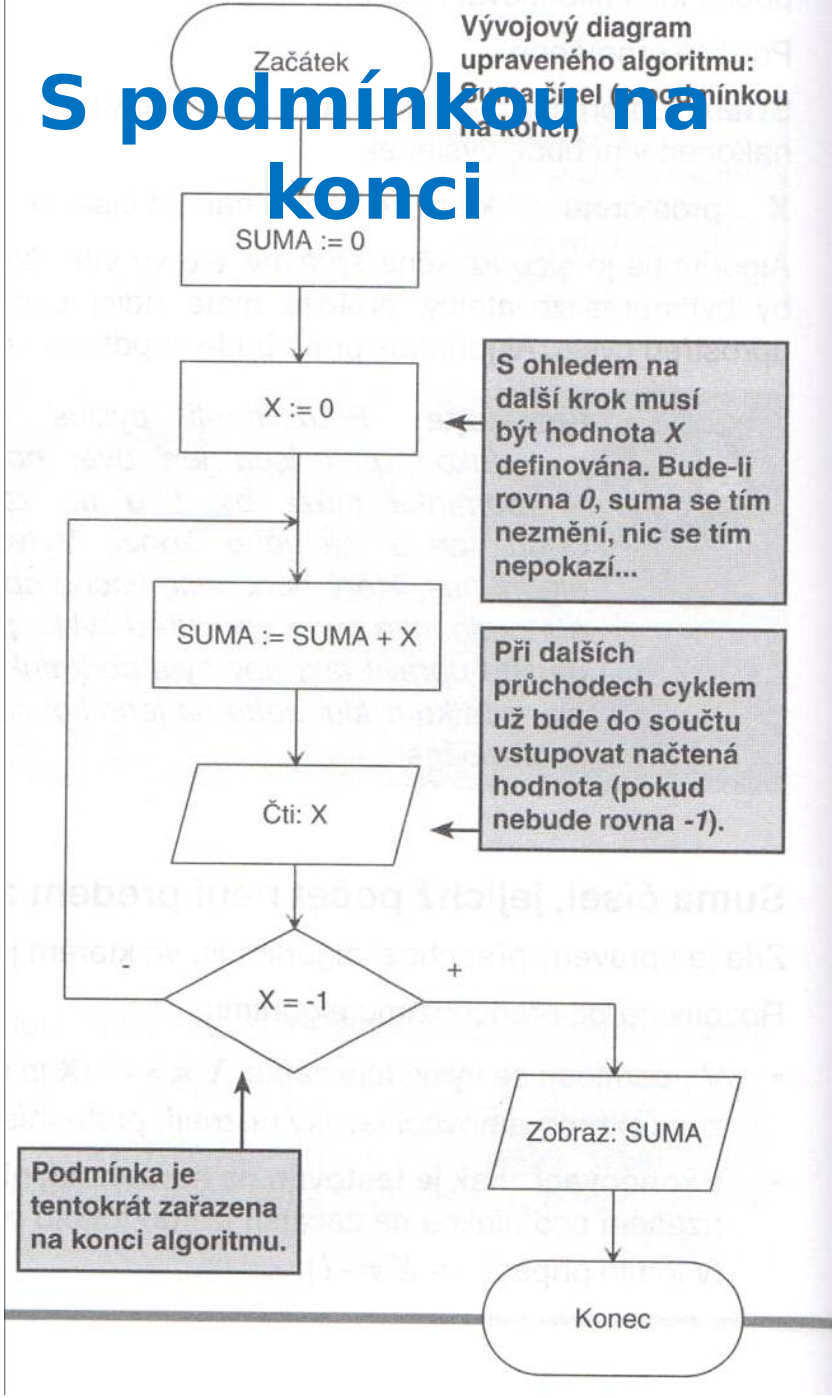
Součet čísel, jejichž počet není předem znám, zadávání čísel končí zadáním -1



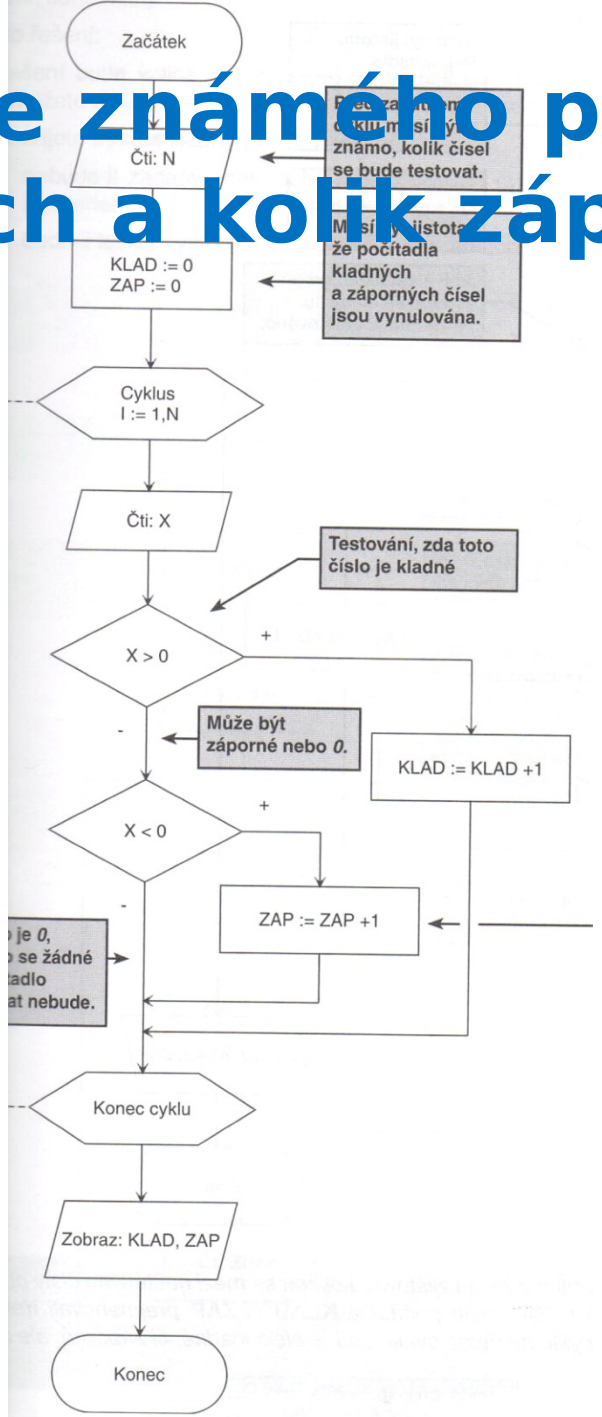
S podmínkou na začátku



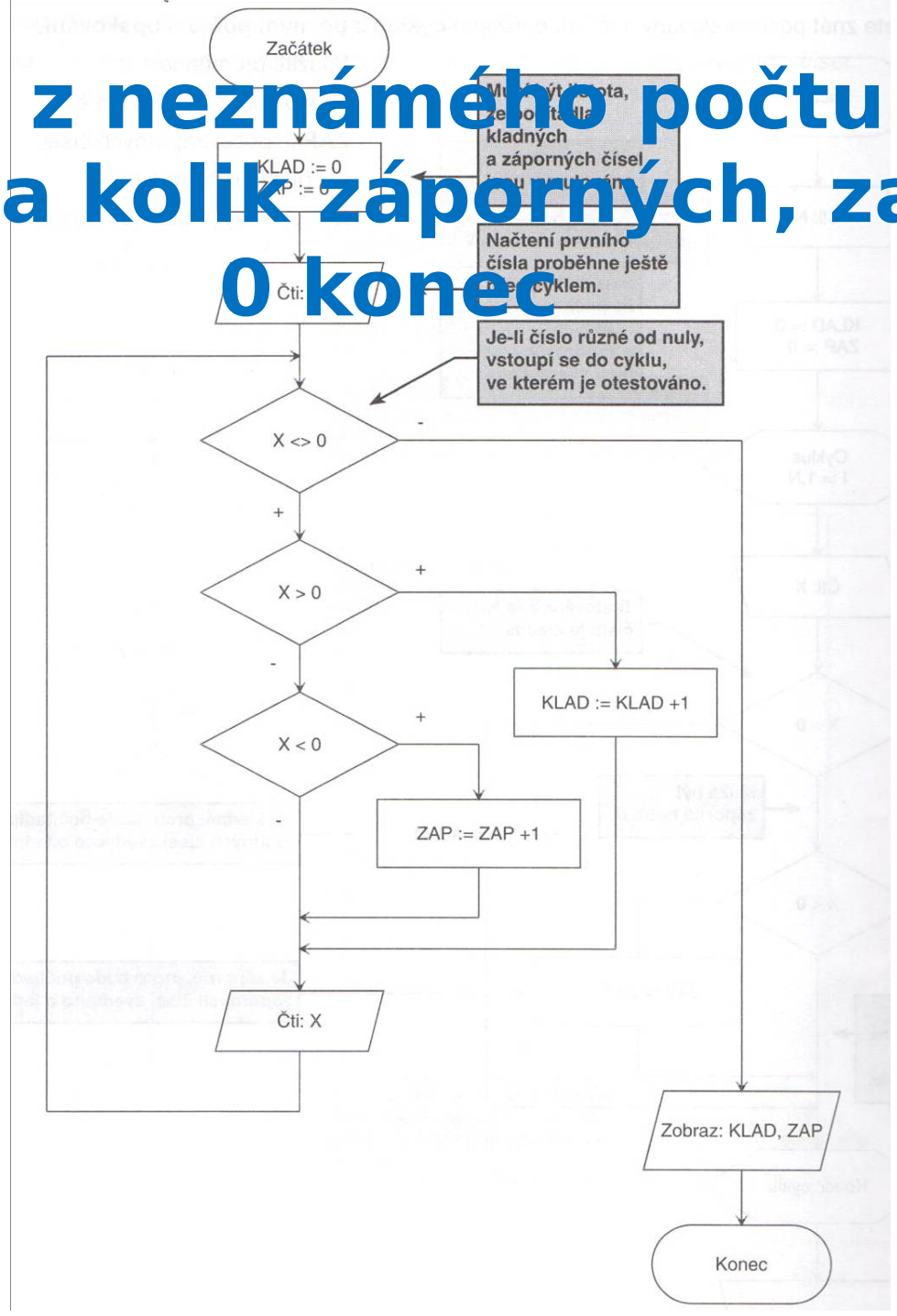
S podmínkou na konci

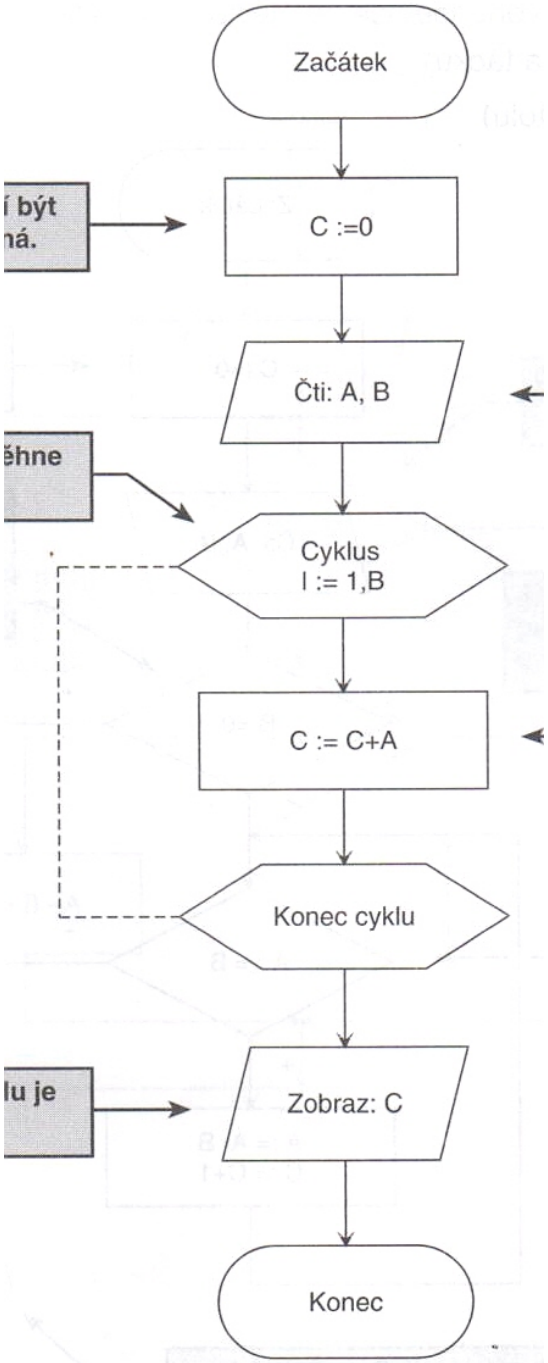


Kolik čísel ze známého počtu čísel je kladných a kolik záporných



Kolik čísel z neznámého počtu čísel je kladných a kolik záporných, zadáním 0 konec





Součin pomocí součtu

5*6 = 5 + 5 + 5 + 5 + 5 +5

8*3 = 8 + 8 + 8

A*B = A+A+A+...+A

A se sečte B –krát

Výsledek uložte do C

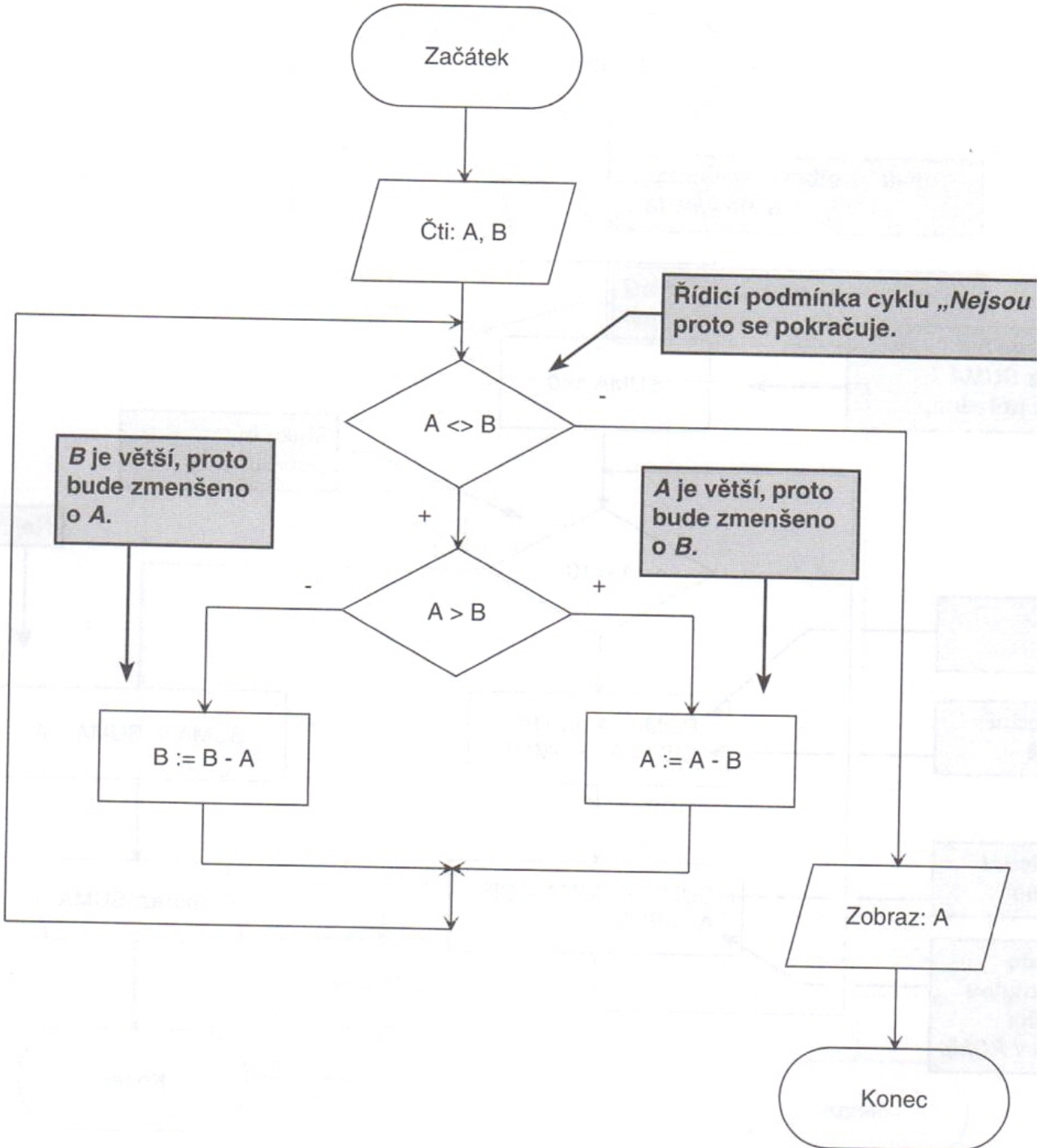
Největší společný dělitel dvou přirozených čísel

Princip řešení:

1. Pokud jsou čísla A a B stejná jsou zároveň svým největším dělitelem
2. Je-li A>B, potom odečteme A-B ….. Nové A
3. Je-li B>A, potom odečteme B-A ….. Nové B
4. Postup opakujeme tak dlouho, dokud A =B

Příklad: A=27 B=12

1. Je-li A>B 27-12=15 …..nové A
2. 15-12=3 …….nové A
3. Je-li B>A 12-3=9 ……..nové B
4. 9-3=6 ……..nové B
5. 6-3=3 ……..nové B
6. Je-li A=B=3 …….největší společný dělitel


Začátek
Čti: A, B
Řídicí podmínka cyklu „Nejsou
proto se pokračuje.
A <> B
-
+
B je větší, proto
bude zmenšeno
o A.
A je větší, proto
bude zmenšeno
o B.
A > B
-
+
B := B - A
A := A - B
Zobraz: A
Konec