

Algoritmizace

Autor: Ing. Jana Pšenčíková
Návrh layoutu: Pavel Navrátil
Zlom a sazba: Ing. Michal Jiříček
Návrh obálky: Ing. Michal Jiříček
Jazyková úprava: PhDr. Dagmar Procházková

© Computer Media s.r.o., 2009
Vydání druhé
Všechna práva vyhrazena

ISBN: 978-80-7402-034-6

Žádná část této publikace nesmí být publikována a šířena žádným způsobem a v žádné podobě bez písemného svolení vydavatele.

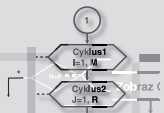
Adresa:
Computer Media s.r.o.
Hrubčická 495
798 12 Kralice na Hané
Česká republika

Telefon: +420 582 302 666
Fax: +420 582 302 667
E-mail: info@computermedia.cz
Web: <http://www.computermedia.cz>

Líbí se Vám tato učebnice? Co v ní postrádáte? Své tipy, postřehy a názory pište na adresu info@computermedia.cz.
Děkujeme Vám.

Obsah

ALGORITMUS	7
CO JE TO ALGORITMUS A PROČ VYTVÁŘÍME ALGORITMY	7
<i>Začátek a konec algoritmu</i>	7
<i>Věcná správnost</i>	9
<i>Jednoznačnost</i>	10
<i>Obecnost</i>	12
<i>Opakovatelnost</i>	12
<i>Srozumitelnost</i>	13
MOŽNOSTI ZÁPISU ALGORITMŮ	13
<i>Slovní vyjádření algoritmu</i>	13
<i>Matematický zápis</i>	14
<i>Rozhodovací tabulky</i>	14
<i>Vývojové diagramy</i>	16
<i>Počítačový program</i>	18
<i>Další formy zápisu algoritmů</i>	18
SEKVENCE	19
<i>Jednoduchá sekvence s jedním sekvenčním blokem – příkaz výstupu</i>	19
<i>Sekvence se dvěma sekvenčními bloky</i>	19
<i>Součet obsahu dvou buněk, přiřazovací příkaz</i>	20
<i>Výměna obsahu dvou buněk</i>	21
<i>Další typické sekvenční algoritmy</i>	22
<i>Rozdíl a součin</i>	22
<i>Obdélník – obvod a plocha</i>	23
<i>Obvod kružnice a plocha kruhu</i>	23
<i>Rovnostranný trojúhelník – obvod a plocha</i>	23
<i>Objem a plocha válce</i>	24
<i>Šestiúhelník - obvod a obsah</i>	24
<i>Výměna hodnot ve dvou buňkách bez pomocné buňky</i>	25
<i>Pythagorova věta</i>	25
VĚTVENÍ	26
OŠETŘENÍ NEŽÁDOUCÍCH DŮSLEDKŮ	26
<i>Křížovátka</i>	26
<i>Podíl</i>	27
<i>Obecnější výraz s dělením</i>	27
<i>Odmocnina</i>	28
<i>Obecný výraz</i>	28
VĚTVENÍ Z DŮVODU NĚKOLIKA ŽÁDOUCÍCH MOŽNOSTÍ	29
<i>Nedělní program</i>	29
VÝRAZY S ABSOLUTNÍ HODNOTOU, VLASTNOSTI ČÍSEL	29
<i>Absolutní hodnota</i>	29
<i>Zjištění, zda je číslo kladné, či záporné</i>	30
<i>Zjištění, zda je číslo sudé, či liché</i>	30
<i>Dělitelnost</i>	31
POROVNÁVÁNÍ A ŘAZENÍ ČÍSEL, MAXIMUM A MINIMUM	32
<i>Porovnání dvou čísel podle velikosti</i>	32
<i>Seřazení dvou čísel s pomocnou buňkou</i>	33
<i>Největší ze tří čísel – bez pomocné buňky</i>	33
<i>Maximum ze tří čísel s použitím pomocné buňky</i>	34
<i>Maximum ze tří čísel s použitím dočasného maxima</i>	34
<i>Seřazení tří čísel podle velikosti bez pomocné buňky</i>	35
<i>Seřazení tří čísel s použitím pomocné buňky</i>	36
<i>Seřazení tří čísel podle velikosti s respektováním výsledku předchozího kroku</i>	37
<i>Seřazení čtyř čísel podle velikosti – s pomocnou buňkou</i>	38
ÚLOHY Z GEOMETRIE	39
<i>Trojúhelník</i>	39
<i>Test na trojúhelník rovnoramenný, rovnostranný, obecný</i>	40
<i>Test na pravoúhlý trojúhelník</i>	41
KOMBINOVANÉ ALGORITMY	43
<i>Lineární rovnice</i>	43
<i>Soustava dvou lineárních algebraických rovnic</i>	43
<i>Kvadratická rovnice s reálnými kořeny</i>	44
<i>Kvadratická rovnice s komplexně sdruženými kořeny</i>	45
<i>Kalkulačka</i>	47
<i>Rychlost, dráha, čas</i>	48
<i>Pohyb rovnoměrně zrychlený</i>	49
<i>Vlaky</i>	50
CYKLY	53
CYKLUS A JEHO TYPY	53



Cykly s pevným počtem opakování	54
Cyklus řízený podmínkou – podmínka je na začátku cyklu	55
Cyklus řízený podmínkou – podmínka je na konci cyklu	56
Čekací smyčka	57
ÚLOHY S OPAKOVÁNÍM	58
Paralelní odpory	58
Kalkulačka	59
SUMY, PROHLÉDÁVÁNÍ ŘADY ČÍSEL, MAXIMUM A MINIMUM	61
Zobrazení čísel od jedničky do desítky	61
Zobrazení čísel od dvojky do dvacítky, jen sudé	61
Suma čísel od 1 do 10	62
Suma 10 různých čísel, resp. libovolného počtu čísel	62
Maximum z deseti kladných čísel	63
Maximum a minimum z celých čísel, jejichž počet bude zadán předem	65
Suma čísel, jejichž počet není předem znám – test uprostřed cyklu	67
Suma čísel, jejichž počet není předem znám – s testem na začátku	67
Suma čísel, jejichž počet není znám předem – s testem na konci cyklu	68
Maximum z neznámého počtu čísel	69
Maximum a minimum z neznámého počtu celých čísel	70
Zjištění, kolik čísel je kladných a kolik záporných – celkový počet je znám	71
Zjištění, kolik čísel je kladných, kolik záporných - přijde-li nula, pak cyklus skončí	72
Zjištění, kolikrát se v textu objeví zadané písmeno – text je zadáván po znacích	73
Zjištění, kolikrát se v textu objeví zadané písmeno – text je zadán najednou	74
Zjištění počtu slov ve větě – znak po znaku	75
Zjištění počtu slov ve větě – načte se celá věta najednou	76
Součin pomocí součtu	77
Dělení pomocí odečítání	78
Největší společný dělitel	79
Součet číslic v čísle	80
Test, kolikrát se vyskytuje určitá číslice v daném čísle	81
ODDECHOVÉ ÚLOHY	82
Písnička „Když jsem já sloužil“	82
Společenská hra „Severní pól“	84
Zající a bažanti	85
ČÍSELNÉ SOUSTAVY A PŘEVODY MEZI NIMI	86
Převod z desítkové soustavy do dvojkové – nové cifry se vypisují od nejvyšších	88
Převod čísla z dvojkové soustavy do desítkové	89
ŘADY – ARITMETICKÉ, GEOMETRICKÉ, DALŠÍ	90
Aritmetická řada – výpočet hodnoty prvků řady, prvky se zobrazují průběžně	90
Aritmetická řada – výpočet hodnoty prvků řady, všechny prvky se zobrazí nakonec	91
Aritmetická řada – součet	92
Geometrická řada – výpočet hodnoty prvků řady, prvky se zobrazují průběžně	93
Geometrická řada – výpočet hodnoty prvků řady, všechny prvky se zobrazí nakonec	94
Geometrická řada – Suma	95
Složitě úrokování	96
Stavební spoření	98
Faktoriál	100
Algoritmus pro výpočet Ludolfova čísla π	101
Algoritmus pro výpočet hodnoty přirozeného logaritmu čísla 2	102
Algoritmus pro výpočet funkce sinus x pomocí mocninné řady	103
Algoritmus pro výpočet funkce e^x pomocí mocninné řady	104
OPERACE S VEKTORY A MATICEMI	105
Součet vektorů	105
Skalární součin vektorů	106
Minimum z řady čísel, metoda přímého výběru – řešení pomocí vektoru	107
Maximum z řady čísel, metoda probublávání – řešení pomocí vektoru	108
Matice – načtení prvků	109
Matice – počet kladných a záporných prvků	110
Největší a nejmenší prvek matice	111
Pozice největšího a nejmenšího prvku matice	112
Trojúhelníková matice	113
Diagonální matice	114
Matice otočená kolem hlavní diagonály	116
Součet matic	118
Součin matic	119
TŘÍDICÍ ALGORITMY	121
Select Sort (třídění přímým výběrem)	121
Bubble Sort (třídění probubláváním) - zjednodušený	123
Bubble Sort (třídění probubláváním) - klasický	124
Shake Sort (třídění přetřásáním)	126

Jednoznačnost

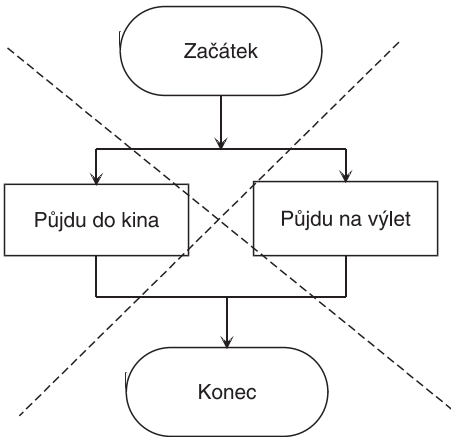
Porušení podmínky jednoznačnosti bývá úplně nejčastější závadou, která se u algoritmů vyskytuje. Je to tím, že tvůrce algoritmu musí pečlivě zvažovat všechny možnosti, které mohou během zpracování nastat. Ukázka opět ozřejmí několik příkladů:

1. Kino nebo výlet

Potkají se dva kamarádi a jeden druhého se ptá: „Co budeš dělat zítra?“

Druhý odpoví: „Půjdu buďto do kina, nebo na výlet.“

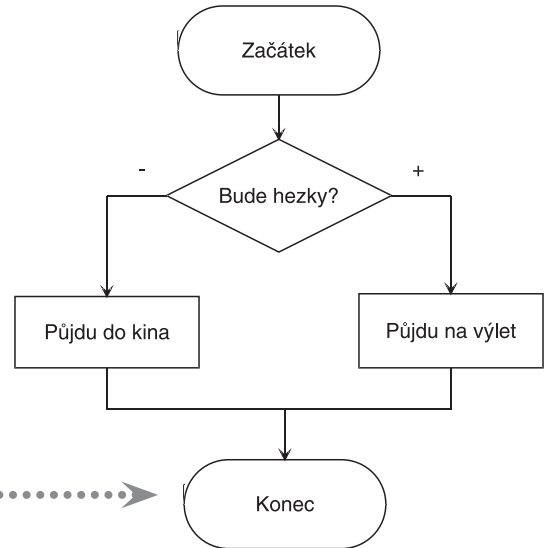
Je to algoritmus?



Je vidět, že v tomto případě je jasně porušena podmínka jednoznačnosti. Dotyčný neví, zda půjde do kina, nebo na výlet.

Člověk by určitě v tomto případě nejednal nahodile, rozhodl by se na základě nějakých dalších okolností, které by nastaly, a rozumově je vyhodnotil.

Pokud by však byl stejným algoritmem řízen robot, pak by nefungoval, protože by se dostal do situace, ve které by se nedokázal rozhodnout.



Aby byl algoritmus správný, musí se do něj doplnit podmínka, na jejímž základě dojde k rozvětvení.

Dialog dvou přátel upravíme:

Potkají se dva kamarádi a jeden druhého se ptá: „Co budeš dělat zítra?“

Druhý odpoví: „Bude-li hezky, půjdu na výlet, nebude-li hezky, půjdu do kina.“

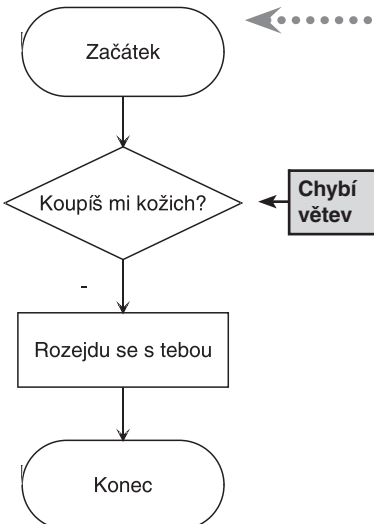
Nyní už je algoritmus správný (viz obr. vpravo).

2. Kožich

Jistý mladík se kamarádí s jednou slečnou. Ta mu řekne: „Nekoupíš-li mi kožich, rozejdu se s tebou.“

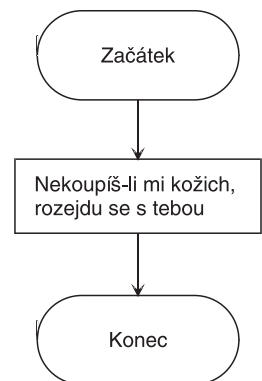
Je možné její tvrzení pokládat za algoritmus (viz obr. vpravo)?

Vidíte, že v tvrzení slečny je skryta podmínka, která však není korektní.



Je zde propracovaná jen jedna větev algoritmu – co se stane, když mladík slečně kožich nekoupí.

Druhá větev algoritmu – když slečna kožich dostane – zde úplně chybí.

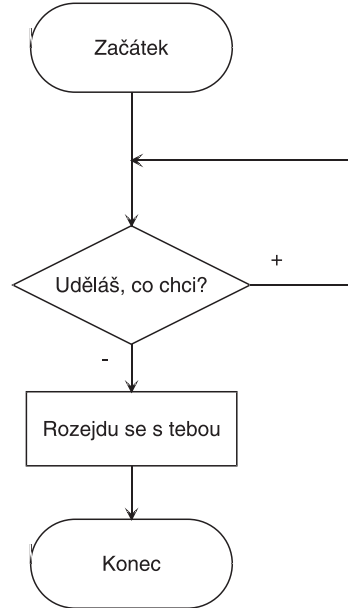
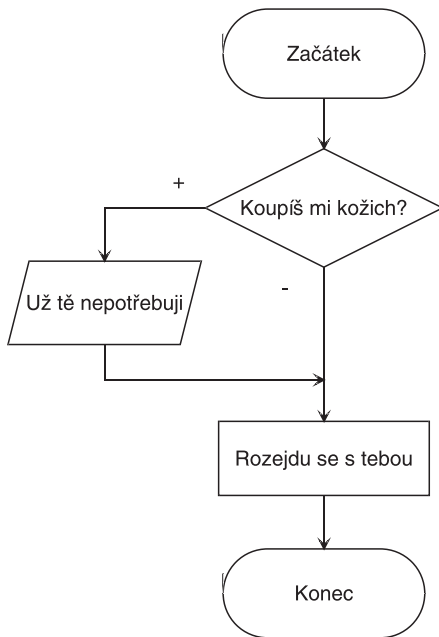


Aby se jednalo o správný algoritmus, musí se doplnit i druhá větev:

V úvahu připadá několik možností:

- dostane-li slečna kožich, zůstane s mladíkem „navěky“ (což je málo pravděpodobné);
- usoudí, že dostala, co chtěla, a s mladíkem se stejně rozejde;
- bude klást další požadavky tak dlouho, až jí jednou mladík nevyhoví, a pak se s ním stejně rozejde.

Takto by vypadaly upravené verze „Už tě nepotřebuji“ a „Kladení dalších požadavků“:



3. Dělení

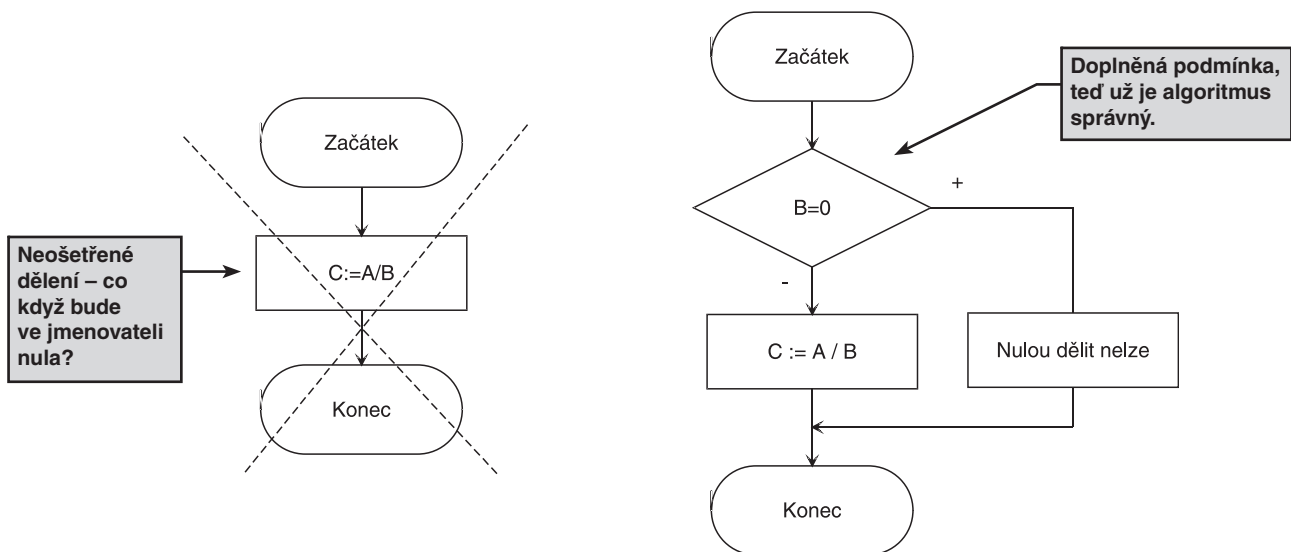
Tento příklad zastupuje celou řadu matematických a logických úloh, ve kterých se musíte zabývat podmínkami řešitelnosti a na které se při algoritmizaci zapomíná. Jsou to zejména:

- **výrazy se zlomky** – jmenovatel se nemůže rovnat nule (pak se hodnota výrazu blíží k nekonečnu);
- **různé další funkce** (například goniometrické, exponenciální, geometrické řady...) – musíte ošetřit oblasti, ve kterých se hodnoty funkcí blíží k nekonečnu;
- **odmocniny** – hledáte-li řešení v oboru reálných čísel, pak výraz pod odmocninou nesmí být záporný.

Na tato ošetření se často zapomíná, a dochází tak k nezávažnějším chybám v programech.

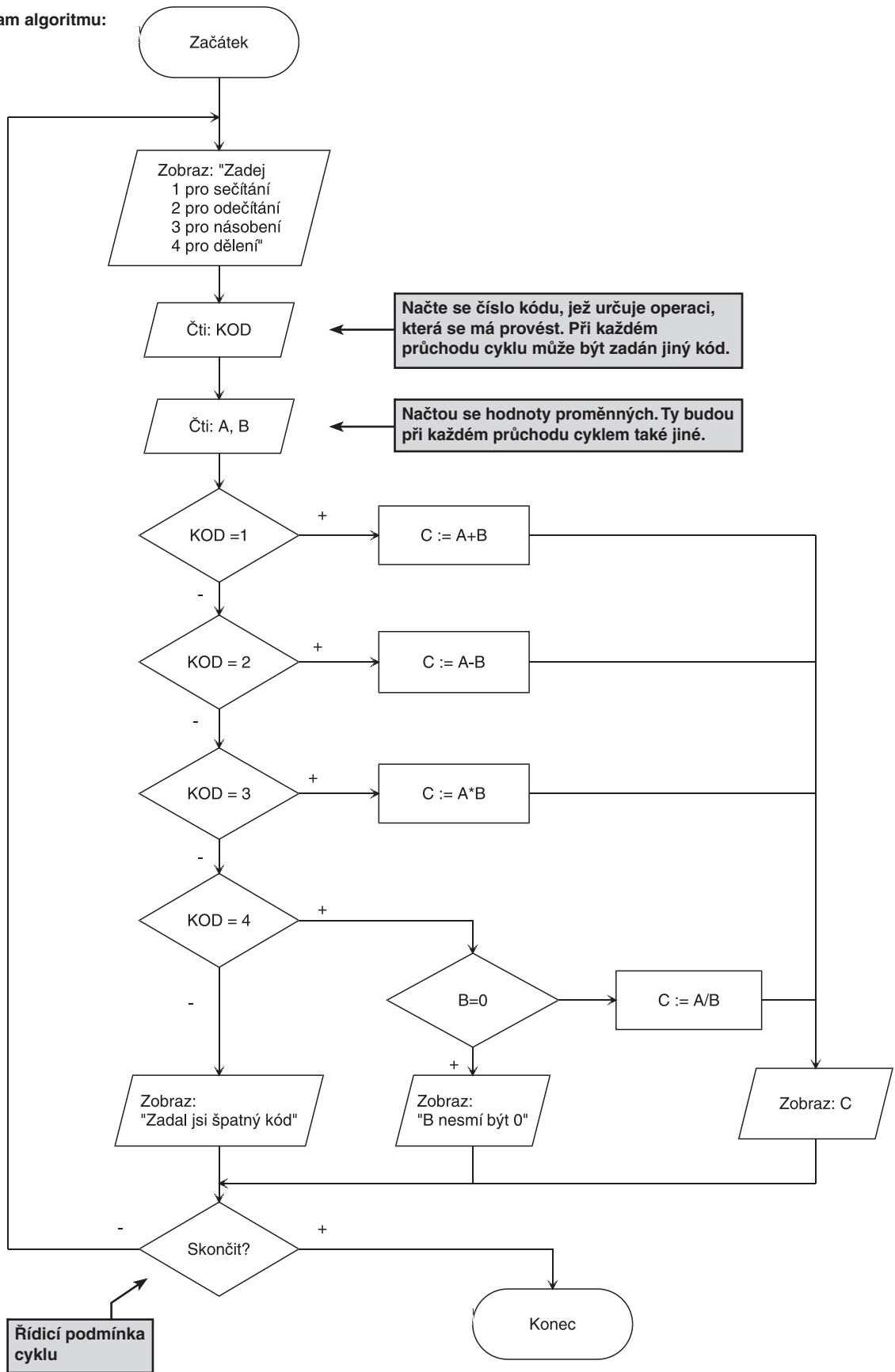
Podmínky řešitelnosti se vždy ošetřují větvením algoritmu:

- na větev, kde je vše v pořádku a pokračuje se ve výpočtu (resp. v jiné činnosti algoritmu)
- na větev, kde se zpravidla uživateli sdělí, že něco brání zdárnému průběhu (jedná-li se už o program, pak se vypíše chybové hlášení) a algoritmus skončí.



Upozornění: Zhotovíte-li program podle špatného algoritmu, ve kterém je porušena podmínka jednoznačnosti, pak program v některých případech (v závislosti na zadaných vstupních datech) pracuje správně a poskytuje správné výsledky. V jiných případech tentýž program (při jiné kombinaci vstupních dat) buď havaruje, nebo se chová navenek korektně, ale poskytuje nesmyslné výsledky.

Vývojový diagram algoritmu:
Kalkulačka



SUMY, PROHLEDÁVÁNÍ ŘADY ČÍSEL, MAXIMUM A MINIMUM

Další velkou skupinou úloh řešených pomocí cyklů jsou:

- různé součty většího počtu čísel;
- úlohy, ve kterých se zjišťuje, zda je ve skupině prvků přítomen určitý prvek (například zda v řadě čísel je přítomno určité číslo, nebo zda je v čísle určitá číslice, písmeno ve větě), popřípadě kolikrát tam je;
- úlohy na vyhledávání maxima a minima z řady čísel.

Pro začátek to bude jednoduchý příklad, na kterém bude vysvětleno, „jak to vlastně funguje“.

Zobrazení čísel od jedničky do desítky

Protože víte přesně, kolik čísel se má zobrazit, použijete **cyklus s pevným počtem opakování**.

Použitá proměnná:

I ... řídicí proměnná cyklu

Při prvním průchodu cyklem má hodnotu **1**.

Při druhém průchodu cyklem má hodnotu **2**.

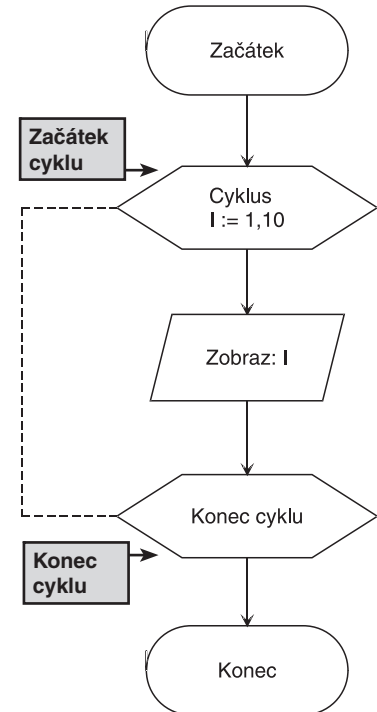
Při třetím průchodu cyklem má hodnotu **3**.

⋮

Při posledním průchodu cyklem má hodnotu **10**.

Při každém průchodu cyklem tuto hodnotu nechejte zobrazit.

Vývojový diagram tohoto algoritmu je na obrázku vpravo.>



Zobrazení čísel od dvojky do dvacítky, jen sudé

Jedná se o mírně modifikovaný předchozí příklad.

Použité proměnné:

I ... řídicí proměnná cyklu

J ... zobrazovaná proměnná, která bude dvojnásobkem řídicí proměnné I.

Všimněte si, že musíte použít novou proměnnou J.



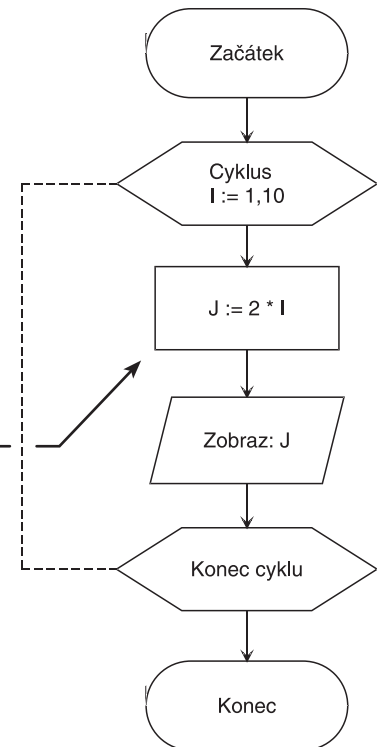
Upozornění: Pozor, v tomto případě nemůžete napsat: $I := 2 * I$, protože by vše sice fungovalo při prvním průchodu cyklem, ale zničili byste řízení cyklu – řídicí proměnná cyklu by nabývala v dalších průchodech úplně jiných hodnot, než by měla!

Musí být použita nová proměnná J



Pamatujte: Řídicí proměnná cyklu musí být použita výhradně k řízení cyklu, nesmíte do ní ukládat žádná jiná data. Může se však vyskytovat na pravé straně přiřazovacího příkazu – lze od ní odvozovat hodnoty jiných proměnných.

Vývojový diagram tohoto algoritmu je na obrázku vpravo.>



Převod z desítkové soustavy do dvojkové – nové cifry se vypisují od nejvyšších

Převeďte číslo z desítkové soustavy do dvojkové, nakonec číslo ve dvojkové soustavě zobrazte.

Princip řešení:

- Výpočet číslic nového čísla ve dvojkové soustavě proběhne úplně stejně jako v předchozím případě.
- Protože však získáváte nižší řády dříve než vyšší, musíte si je někde ukládat. K tomuto účelu použijete **strukturovanou proměnnou pole**. Toto pole bude mít několik jednoduchých prvků, např.: **A[1]** je první prvek pole, **A[2]** je druhý prvek pole, **A[I]** bude I-tý prvek pole, atd. Do každého prvku tohoto pole si můžete uložit nějaké číslo – v tomto případě to bude číslice nového čísla ve dvojkové soustavě.
- Na závěr všechny prvky pole vypíšete, začnete posledním a skončíte prvním. Použijete k tomu **cyklus s pevným počtem opakování**, protože už víte, kolik prvků pole má (to si zjistíte, když budete při výpočtu číslic počítat průchody cyklem).

Použité proměnné:

X ... číslo v desítkové soustavě zadané zvenčí

Y ... zbytek po celočíselném dělení dvěma, jedna cifra nového čísla ve dvojkové soustavě

POM ... pomocná buňka, do které se ukládá výsledek celočíselného dělení dvěma

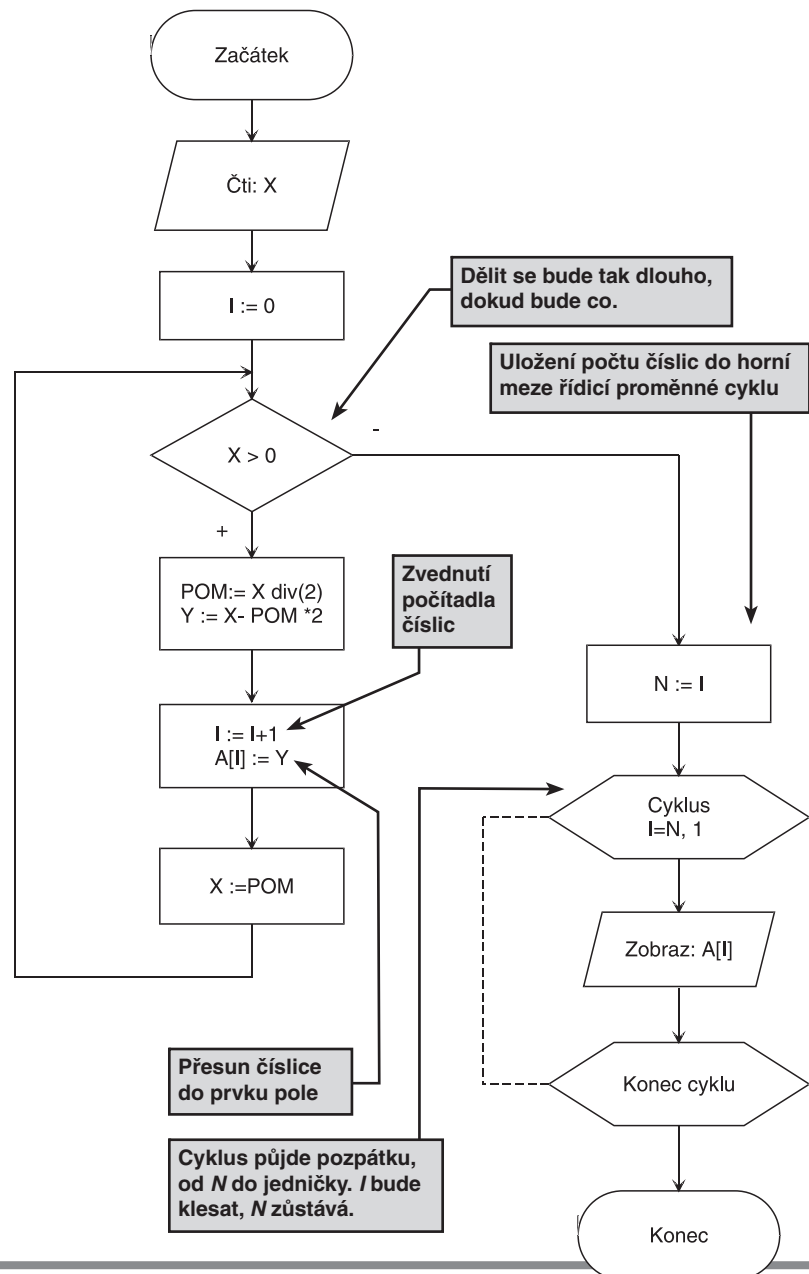
A[I]... I-tý prvek pole **A**, do kterého se budou ukládat spočítané číslice

I ... počítadlo číslic (při každém průchodu cyklem se získá jedna číslice a do počítadla se přidá jednička), v druhém cyklu bude počítadlo použito jako řídicí proměnná cyklu



Poznámka: Proměnnou **Y** byste mohli vynechat a zbytky celočíselného dělení načítat zrovna do prvku pole **A[I]** – bylo by to úspornější. Tato verze byla ponechána kvůli lepší srozumitelnosti a návaznosti myšlenkových pochodů na předchozí algoritmus.

V první části algoritmu (při výpočtu jednotlivých číslic) je použita modifikace předchozího algoritmu doporučená v Tipu.



TŘÍDICÍ ALGORITMY

Pověstnou „třešničkou na dortu algoritmizace“ jsou **třídící algoritmy**. Jejich úkolem je vždy seřadit podle velikosti řadu číselných nebo alfanumerických údajů.

Bez třídících programů se neobejde žádná úloha hromadného zpracování dat, neboť vyhledávání informací v seřazených datech je podstatně jednodušší a rychlejší než v datech neseřazených (kde připadá v úvahu pouze sekvenční prohledávání).

Načtená data musí být vždy před vlastním tříděním uložena, a to buď v operační paměti, nebo na disku, **protože se ke každé položce budete několikrát vracet**.



Upozornění: Nemůžete použít systém jediné proměnné, do které budete zvenčí zadávat údaje a kterou vždy v dalším kroku cyklu přepíšete novým údajem!

A) Z pohledu zadání vstupních dat existují dvě varianty:

- **setřídění předem známého počtu čísel či znaků** - počet je znám předtím, než se načte první číslo či znak;
- **setřídění neznámého počtu čísel** - načítají se čísla či znaky a na jejich konci je „zarážka“ (může to být buď speciální znak, který se normálně v řadě tříděných znaků či čísel nemůže vyskytnout, nebo například značka konce souboru).

Tyto varianty mají vliv pouze na „přípravnou fázi“ třídícího algoritmu – na způsob načtení a uložení vstupních dat (zda se použije cyklus s pevným počtem opakování, či cyklus řízený podmínkou). Na vlastní průběh třídění však vliv nemají.

B) Z pohledu vlastního způsobu seřazování dat existuje hodně různých metod, které se liší:

- složitostí algoritmu;
- nároky na paměť (počtem pomocných proměnných);
- rychlostí (počtem porovnání).

V učebnici bude objasněno několik nejznámějších třídících algoritmů.



Poznámka: Ve všech třídících algoritmech se na začátku vyskytuje načtení údajů, které je stejné pro všechny uváděné algoritmy. Tato část již nebude rozváděna, neboť se už v této knize vyskytuje několikrát. Je označena značkou podprogramu **Načti vektor A[N]** a rozumí se tím načtení n čísel či znaků do strukturované proměnné pole **A** o n položkách.

Select Sort (třídění přímým výběrem)

Název metody **Select Sort** pochází z toho, že postupně vybíráte jeden prvek po druhém a srovnáváte jej s prvním prvkem neseříděné posloupnosti. Je-li prvek, který je „na řadě“, menší než první, vyměníte je.

První setkání s tímto algoritmem bylo už v kapitole **Větvení – podkapitola Porovnávání čísel a jejich řazení podle velikosti**, příklad **Seřazení tří čísel s použitím pomocné buňky**, resp. **Seřazení čtyř čísel s použitím pomocné buňky**. Nyní je algoritmus zobecněn na obecný počet čísel či znaků a je řešen pomocí cyklů.



Poznámka: V této úloze bude použito třídění vzestupné, tj. od nejmenšího prvku k největšímu. Pokud byste chtěli třídít naopak, postup by byl analogický, pouze byste namísto minima hledali maximum.

Princip řešení:

- Načtete všech n položek do pole **A** – viz podprogram **Načti vektor A[N]**.
- Položky setřídíte takto:
 - Nejprve ze všech n položek najdete minimum a uložíte je do položky **A[1]**. K vyhledání minima použijete algoritmus z této kapitoly **Minimum z řady čísel, metoda přímého výběru - řešení pomocí vektoru** (str. 107).