Vlastnosti algoritmů, notace, zápis

Jitka Kreslíková, Aleš Smrčka

2021

Fakulta informačních technologií
Vysoké učení technické v Brně

IZP – Základy programování

Vlastnosti algoritmů, notace, zápis

- Co je algoritmus?
- Vlastnosti algoritmů
- Vyjadřování algoritmů

Co je algoritmus?

Algoritmus je konečná, uspořádaná množina úplně definovaných pravidel pro vyřešení nějakého problému.

ČSN ISO/IEC 2382-1, Informační technologie - Slovník - Část 1: Základní termíny

- postup, který nás dovede k řešení úlohy
- jedná se o přesně definovanou konečnou posloupnost příkazů (kroků), jejichž prováděním pro každé přípustné vstupní hodnoty získáme po konečném počtu kroků odpovídající hodnoty výstupní
- slovo „algoritmus”: je odvozeno ze jména arabského učence, který se jmenoval Muhammad ibn Musa Abdallah Al Khowarizmi (Chorezmí) a žil na přelomu 8. a 9. století na území dnešního Uzbekistánu. Zasloužil se zejména o rozšíření algoritmů pro aritmetické operace v poziční soustavě (prakticky vytvořil systém arabských číslic).

Co je algoritmus?

Základní pojmy:

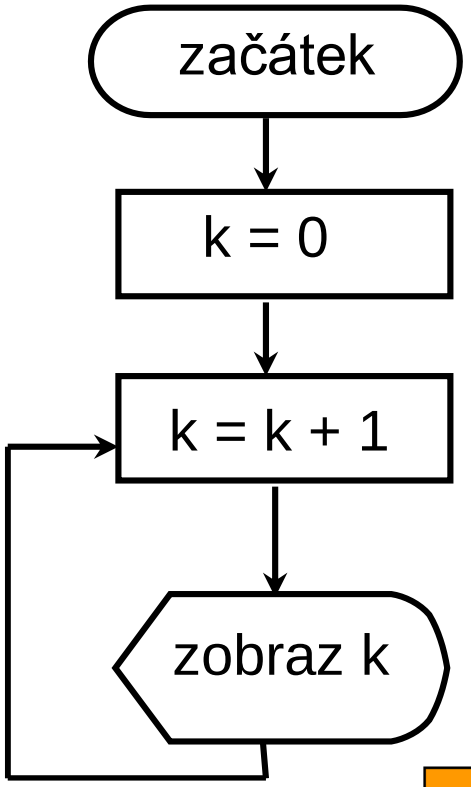
- **Informace** je poznatek (týkající se jakýchkoliv objektů, např. fakt, událostí, věcí, procesů nebo myšlenek, včetně pojmů), který má v daném kontextu specifický význam.
- **Data** jsou opakovaně interpretovatelná formalizovaná podoba informace vhodná pro komunikaci, vyhodnocování nebo zpracování.
- **Program** je jednoznačný předpis, podle kterého je počítač schopen provádět výpočty nějakého algoritmu.
- **Procesor** je prvek, kterému je svěřeno vykonávání algoritmu.

Vlastnosti algoritmů

- **konečnost** (rezultativnost) - má zaručit vyřešení úlohy po konečném počtu kroků
- **hromadnost** - jedním algoritmem lze řešit celou třídu úloh stejného druhu
- **determinovanost** – algoritmus je zadaný ve formě konečného počtu jednoznačných pravidel
- **efektivnost** - na správný průběh programu nemá žádný vliv, zajišťuje pouze to, aby program trval co nejkratší dobu

Vlastnosti algoritmů

Příklad 1: konečnost



vstup: 0
výstup: posloupnost čísel 1, 2, 3, ….

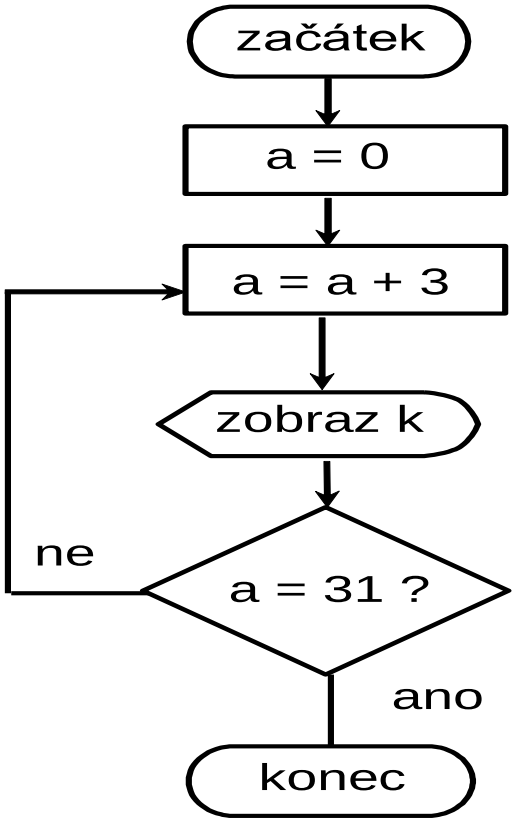
algoritmus není **konečný**.

navíc chybí konec

Vlastnosti algoritmů

Příklad 2: konečnost

Algoritmus, který skončí až se proměnná "a" bude rovnat 31



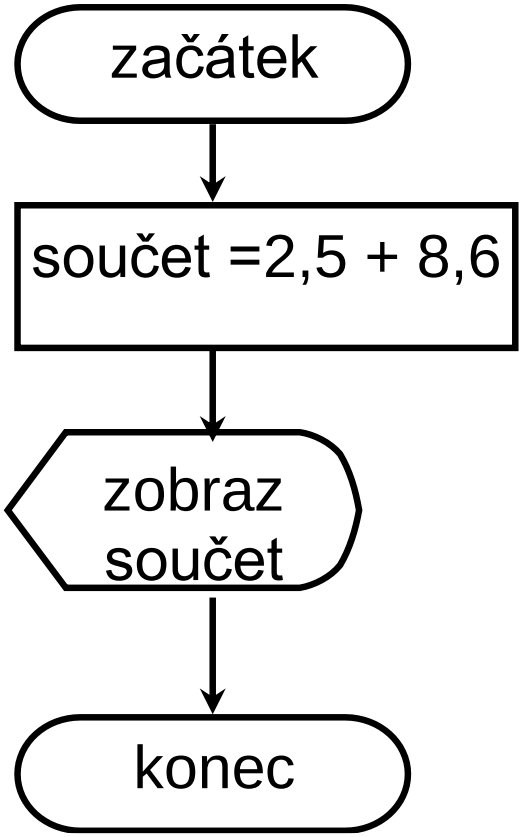
vstup: 0
výstup: posloupnost čísel 3, 6, …….

algoritmus není **konečný**

algoritmus je **zacyklený**

Vlastnosti algoritmů

Příklad 3: hromadnost



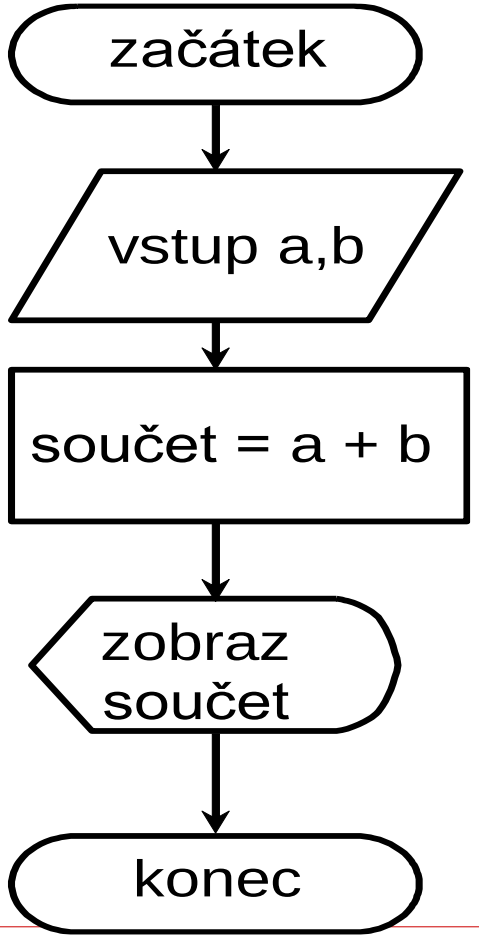
vstup: 2,5 8,6
výstup: součet čísel

algoritmus není **hromadný**

sečte pouze dvě konkrétní čísla.

Vlastnosti algoritmů

Příklad 4: hromadnost

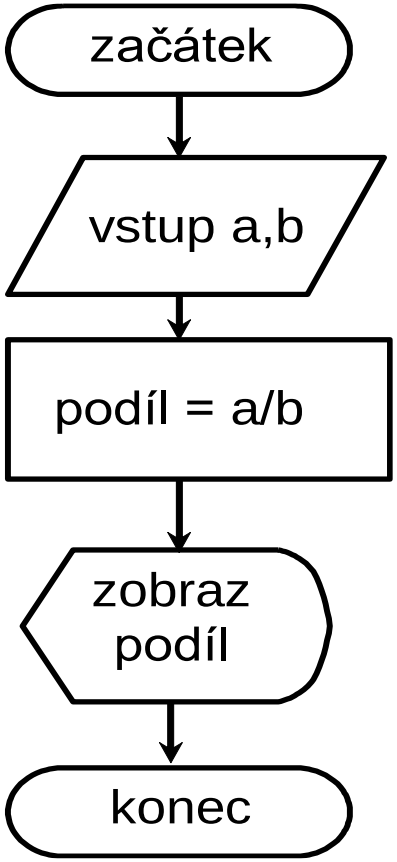


algoritmus je **hromadný**

sečte libovolná dvě zadaná čísla

Vlastnosti algoritmů

Příklad 5 : determinovanost



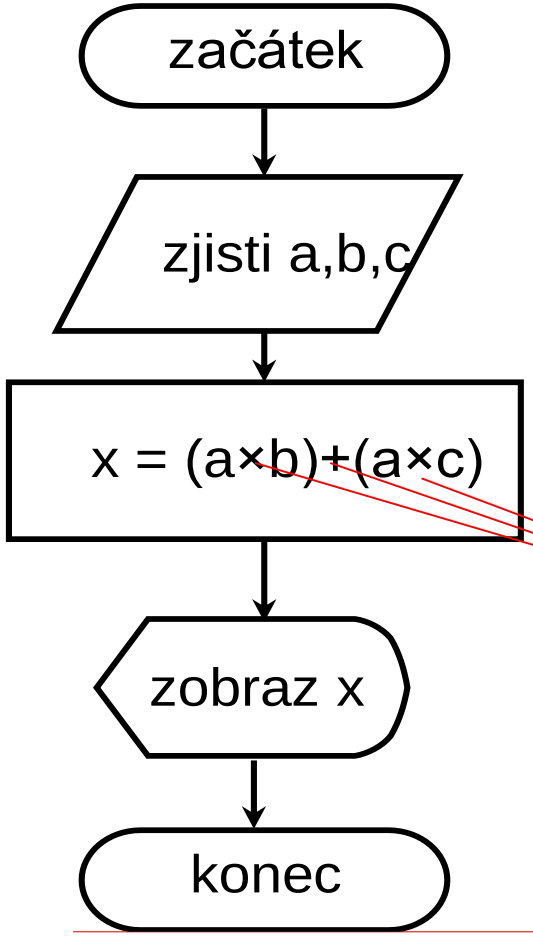
vstup: 2 celá čísla
výstup: podíl zadaných čísel

algoritmus **není determinovaný**

co když b = 0?.

Vlastnosti algoritmů

Příklad 6: efektivnost



vstup: a, b, c
výstup: x = (a×b)+(a×c)

algoritmus **není efektivní**

3 operace

vhodnější je upravit na:
x = a × (b+c)

Vyjadřování algoritmů

- Slovní popis
- Vývojové diagramy
- Struktogramy
- Rozhodovací tabulky
- Programovací jazyky

Slovní popis

Příklad: Je dáno přirozené číslo n (n > 2). Máme zjistit, zda je toto číslo prvočíslem.

V intervalu <2,n-1> nesmí existovat žádný dělitel čísla n.

1. zvolíme hodnotu n
2. přiřadíme i=2
3. je-li i dělitelem n, jdeme na krok 7
4. i zvýšíme o 1, i = i+1
5. je-li i menší než n, jdeme na krok 3
6. signalizujeme, že n je prvočíslo, jdeme na krok 8
7. signalizujeme, že n není prvočíslo
8. konec výpočtu

Vývojové diagramy

ČSN ISO 5807 Zpracování informací. Dokumentační symboly a konvence pro vývojové diagramy toku dat, programu a systému, síťové diagramy a diagramy zdrojů systému. (1996)

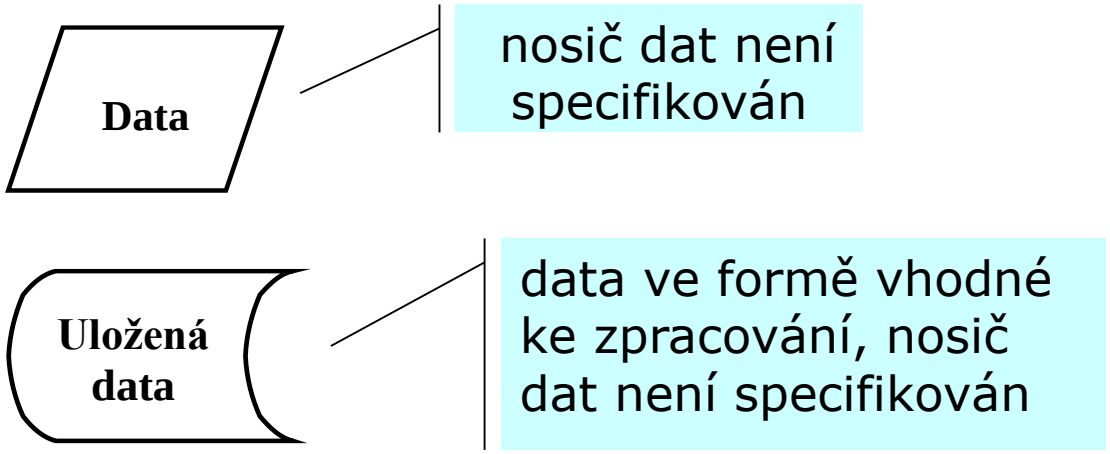
Vývojové diagramy zobrazují tok dat a posloupnost operací v programu.

Obsahují:

- **symboly** pro vlastní operace zpracování včetně symbolů definujících stanovený tok, který má být dodržen při zachování logických podmínek
- **symboly spojení** indikují tok řízení
- **zvláštní symboly** pro usnadnění čtení a zápisu vývojového diagramu

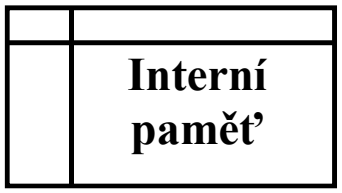
Symboly dat

- **základní symboly dat**



Symboly dat

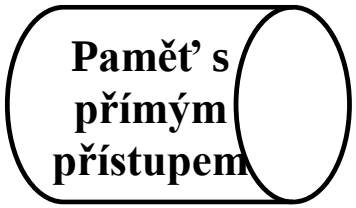
specifické symboly dat



nosičem dat je vnitřní paměť

data přístupná pouze sekvenčně, kde je nosičem dat např.

- magnetická páska
- magnetickopáskový zásobník
- magnetickopásková kazeta



data přímo přístupná, nosič dat je např.

- magnetický disk
- magnetický buben
- pružný disk

Symboly dat

pro člověka přímo čitelná data např.
- tištěný výstup, OCR nebo MICR dokument
- mikrofilm
- kotouč pokladní pásky
- formulář pro vstup dat

pro ruční vstup informací v době zpracování např.
- spřažená klávesnice
- nastavené přepínače
- světelné pero
- snímač čárového kódu

nosičem dat jsou štítky např.
- děrné štítky
- magnetické štítky
- značkové děrné štítky
- štítky s útržkem
- štítky (platební karty) s optickým snímáním značek

Symboly dat

nosičem dat je papírová páska

nosič dat, na kterém se zobrazují informace použitelné lidmi např.

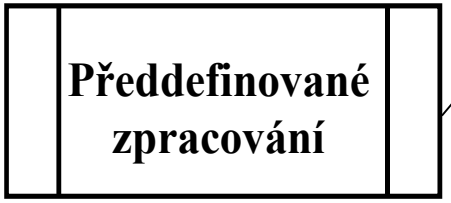
- obrazovky
- spřažené indikátory

Symboly zpracování

základní symbol zpracování

jakýkoliv druh funkce zpracování, např.
- provádění definované operace
- skupina operací, jejichž výsledkem je změna hodnoty, formy nebo umístění informací
- stanovení, který z několika směrů toků se má sledovat

specifické symboly zpracování



pojmenované zpracování, které se skládá z jedné nebo více operací nebo programových kroků, jež jsou specifikovány jinde, např.
- podprogram
- modul

Symboly zpracování

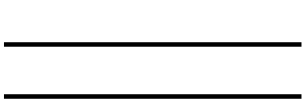
jakékoliv zpracování uskutečňované člověkem

úprava instrukce nebo skupiny instrukcí pro ovlivnění určité následné činnosti, např.

- nastavení spínače
- úprava indexového registru
- inicializace podprogramu

rozhodovací nebo přepínací funkci s jedním vstupem, kde však může být řada alternativních výstupů, z nichž pouze jeden může být aktivován po vyhodnocení podmínek definovaných uvnitř symbolu

Symboly zpracování



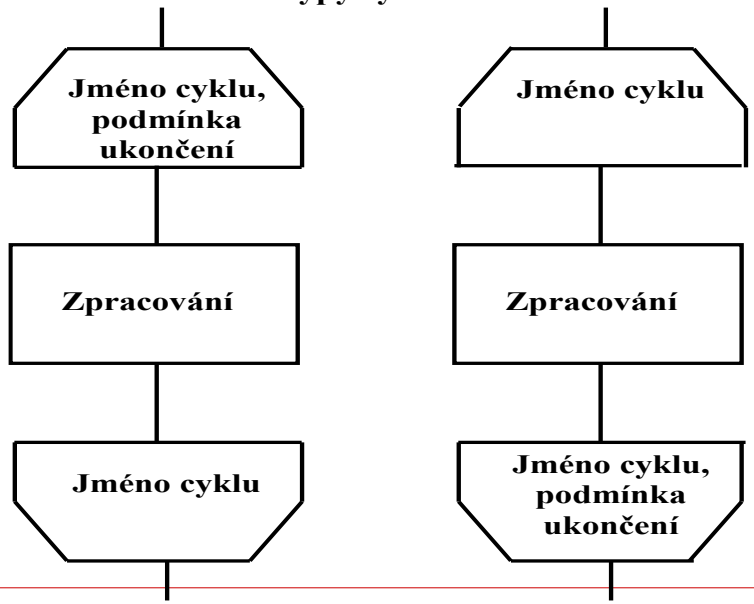
Paralelní režim

synchronizace dvou nebo více operací

začátek a konec cyklu
- obě části symbolu mají stejný identifikátor
- podmínky pro inicializaci, načítání a ukončení

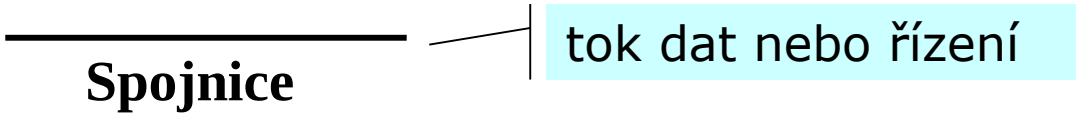
se objevují uvnitř symbolu na začátku nebo na konci podle umístění testovací operace

Typy cyklů:

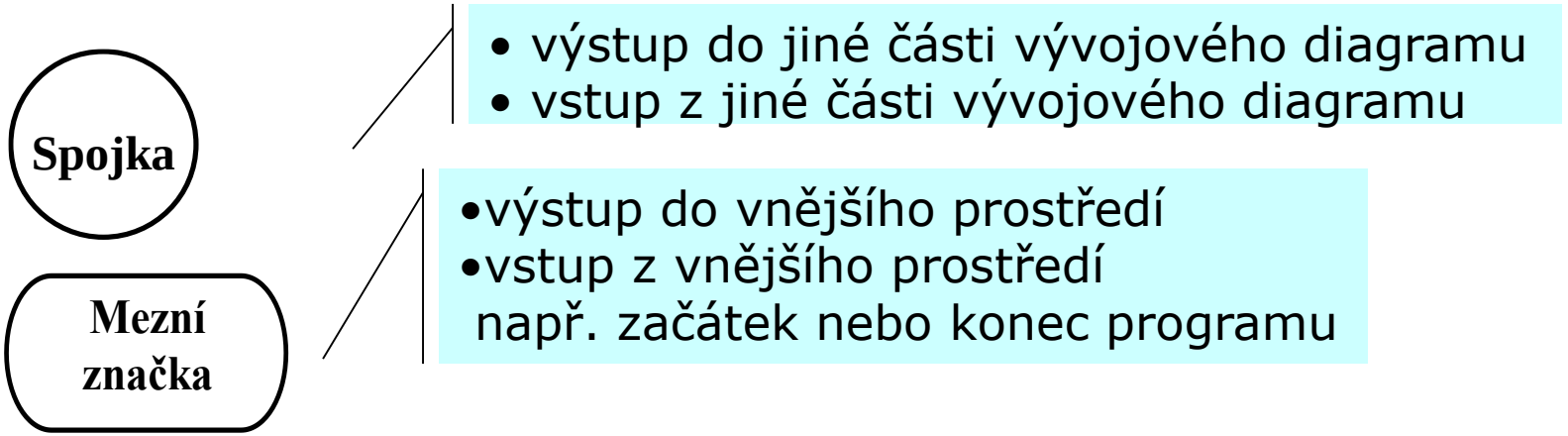


Symboly spojení

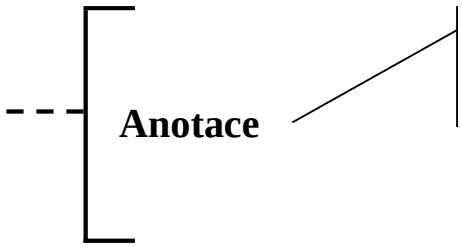
základní symbol spojení



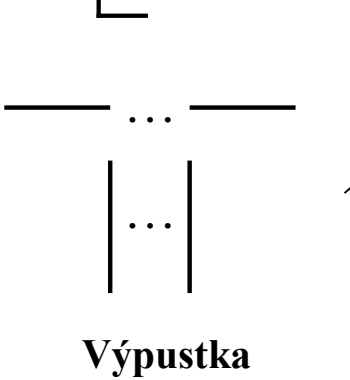
zvláštní symboly



Symboly spojení

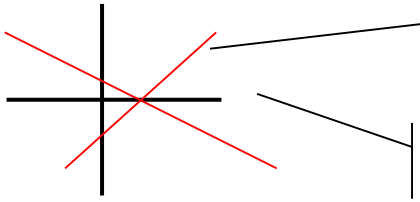


- připojení popisných komentářů
- připojení vysvětlujících poznámek



vypuštění symbolu nebo skupiny symbolů, kde ani druh ani počet symbolů nemusí být definován

Křížení spojnic:

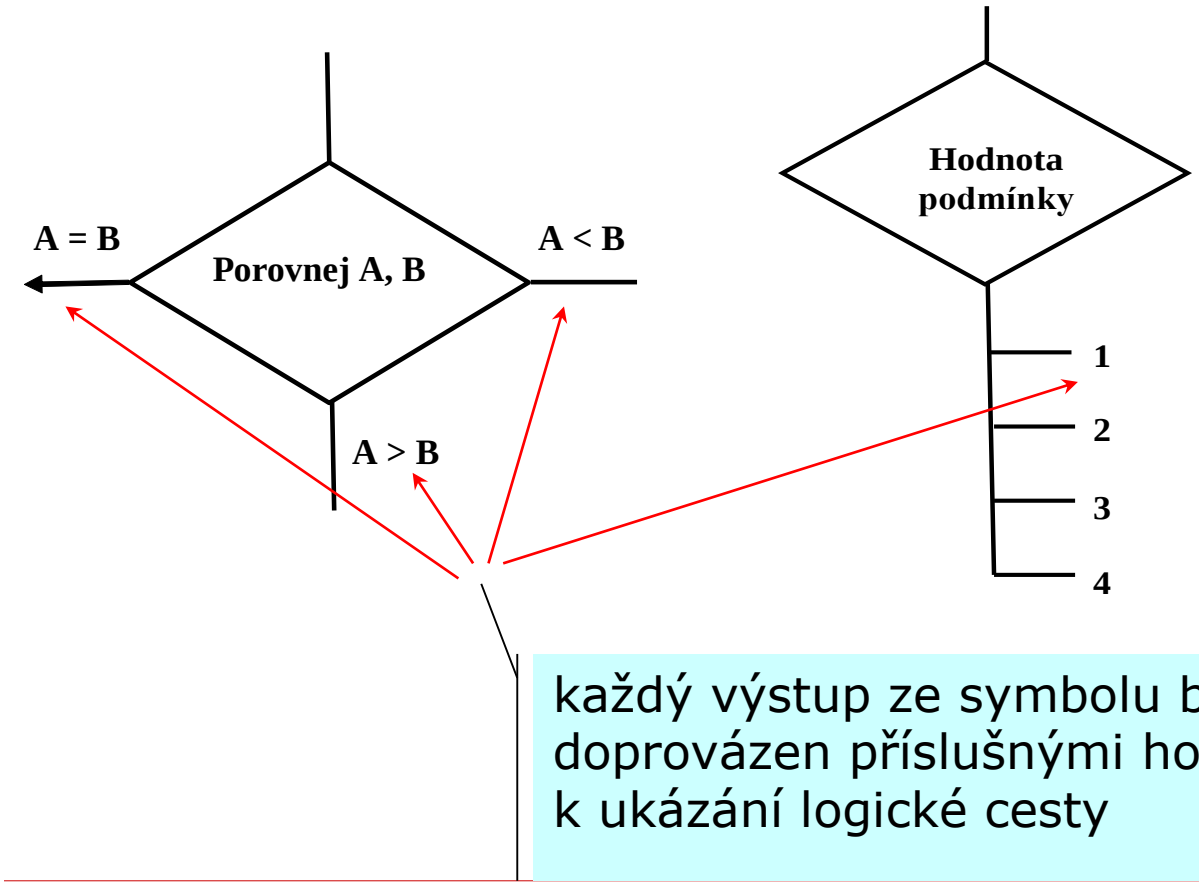


se nedoporučuje

změny směru v bodech křížení nejsou možné

Symboly spojení

Vícenásobné výstupy:



každý výstup ze symbolu by měl být doprovázen příslušnými hodnotami podmínek k ukázání logické cesty

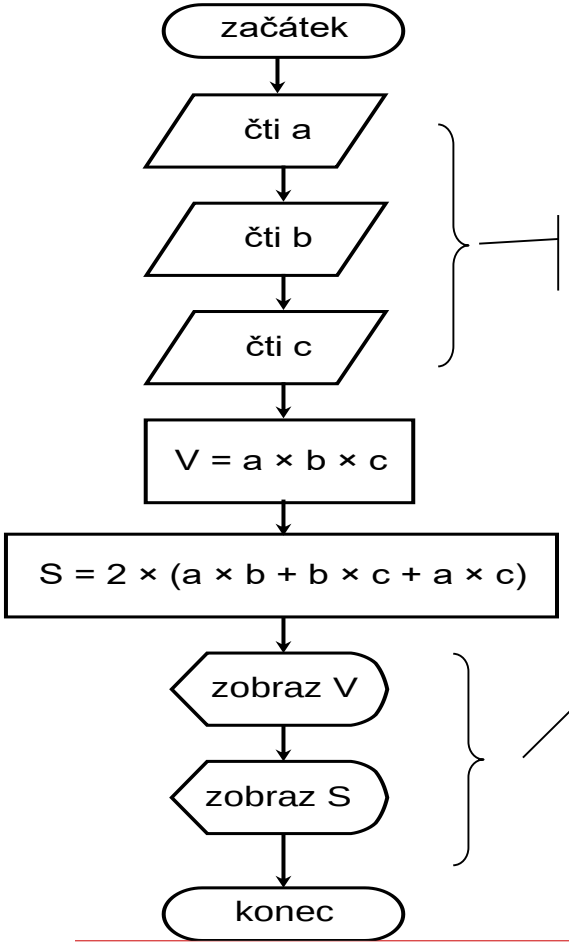
Vývojové diagramy - příklady

Algoritmus lze zaznamenat kombinací tří řídicích struktur:

- **sekvence** (posloupnost)
- **selekce** (větvení)
- **iterace** (cyklus)

Sekvence

Příklad 1: sekvence



vstup: 3 celá kladná čísla (rozměry kvádru)
výstup: 2 celá čísla (objem, povrch kvádru)

načti vstupní data do paměti, např. z klávesnice

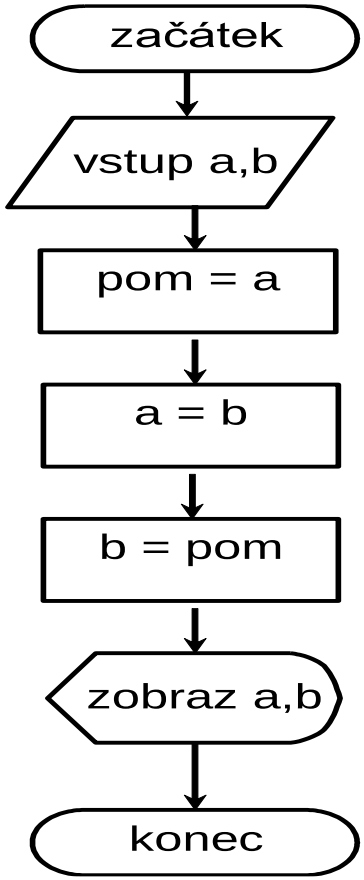
zpracuj data v paměti

zobraz data z paměti na monitor

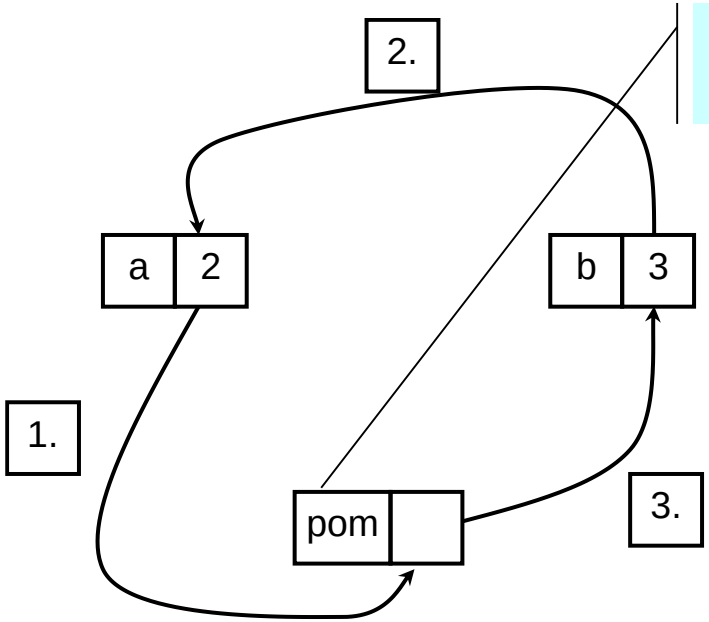
Sekvence

Příklad 2: sekvence

Sestavte algoritmus pro záměnu obsahu dvou proměnných a, b.



vstup:	a	b
	2	3
výstup:	a	b
	3	2

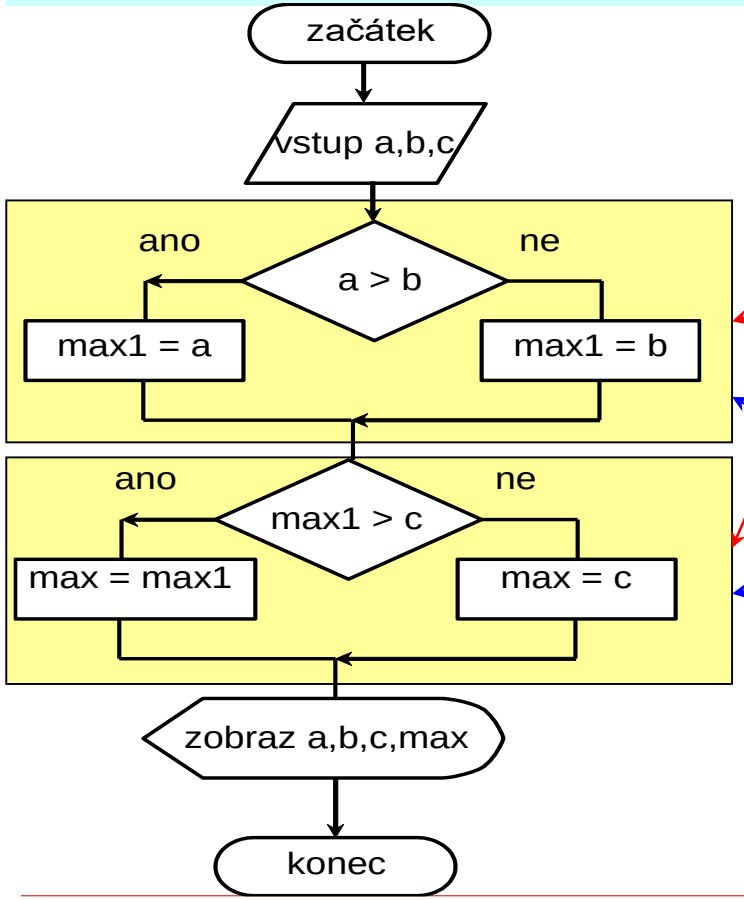


použijeme pomocnou proměnnou pom

Selekce

Příklad 3: selekce

Jsou dána tři různá celá čísla a, b, c. Určete, které z nich je největší.



vstup: a b c
výstup: a b c max (a, b, c)

jeden vstup
jeden výstup

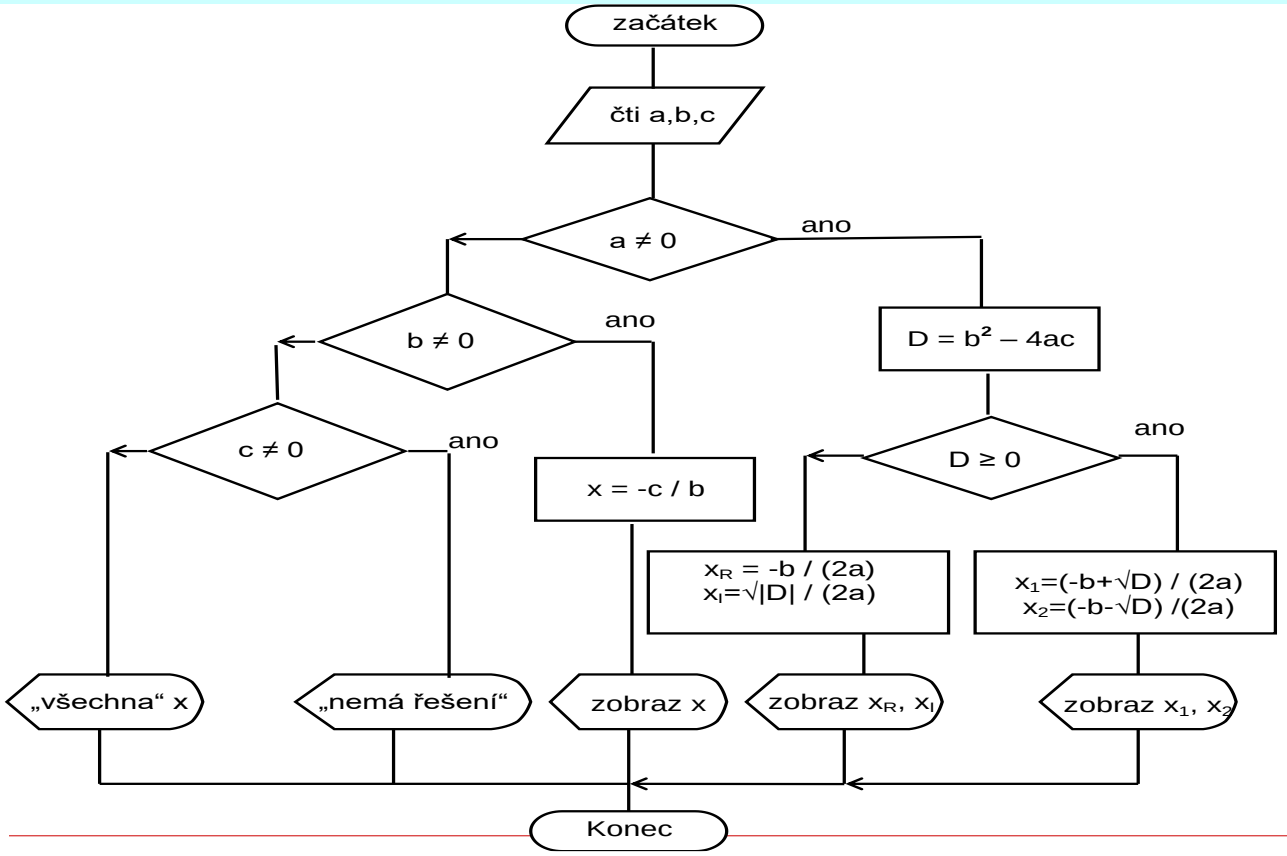
dvě shodné konstrukce

max (x,y)

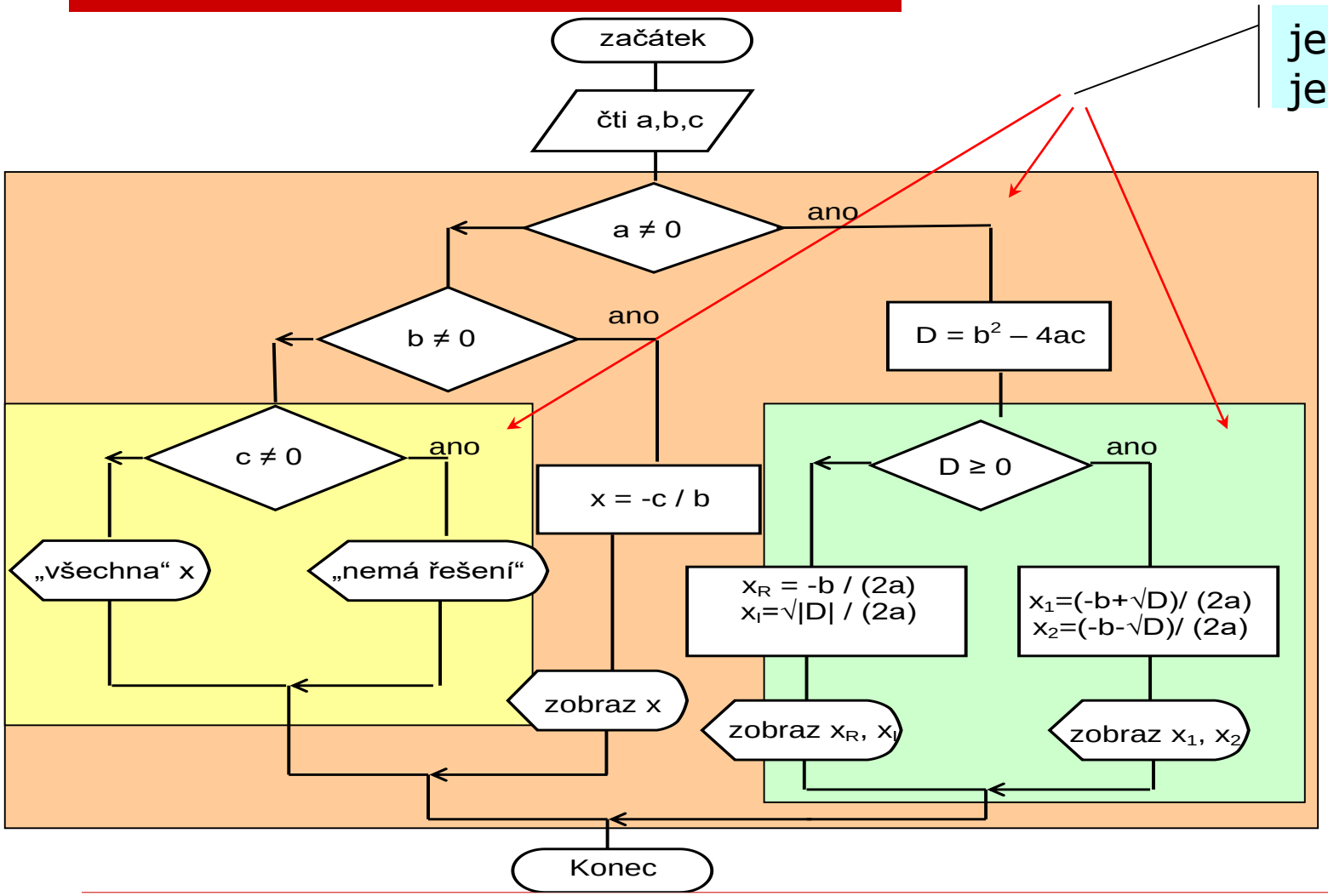
Selekce

Příklad 4: selekce

Sestavte algoritmus pro výpočet kořenů kvadratické rovnice, která je definována ve tvaru: $ax^2 + bx + c = 0$.



Vývojové diagramy - příklady

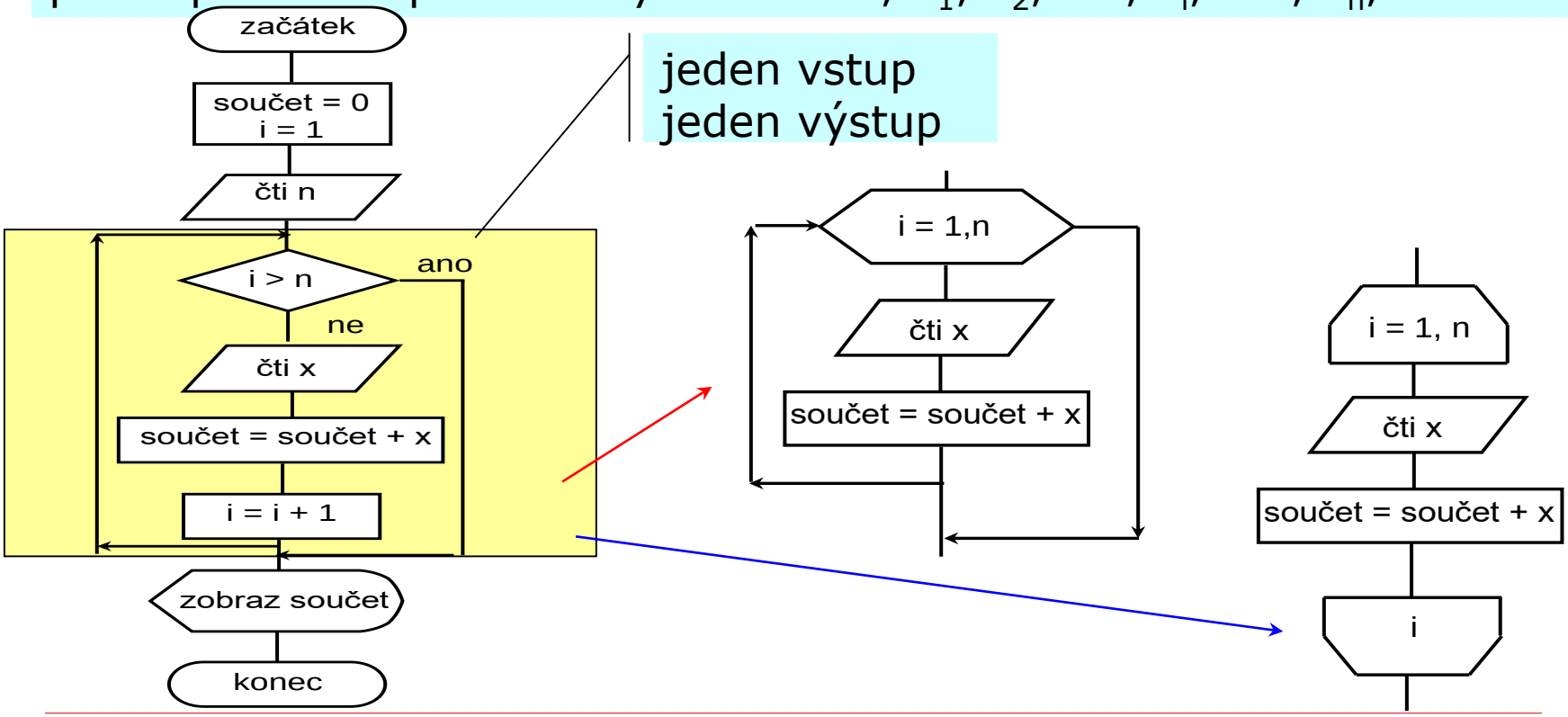


jeden vstup
jeden výstup

Iterace

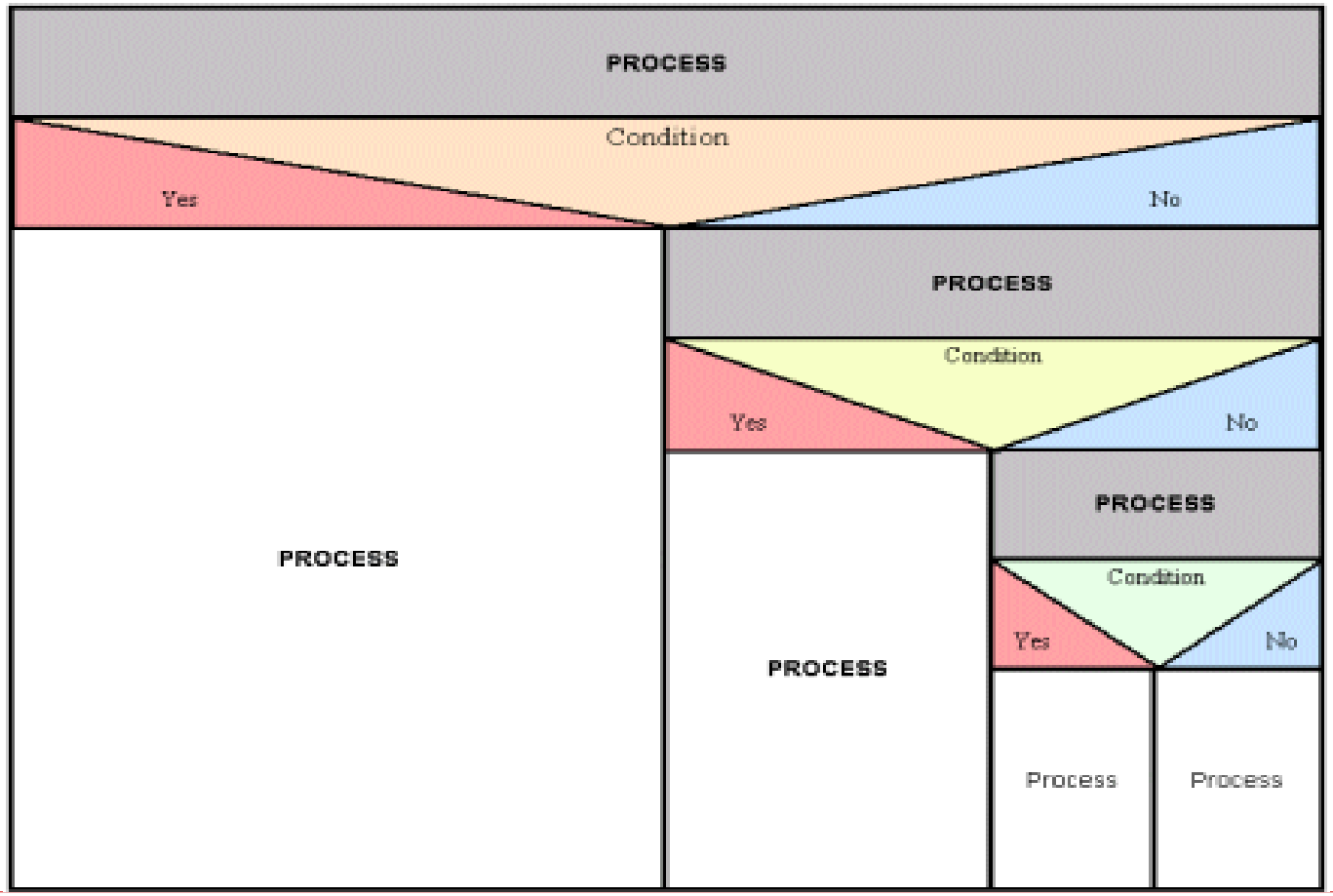
Příklad 5: iterace

Čtěte posloupnost celých čísel a zjistěte jejich součet. Počet čísel v posloupnosti je znám a jeho hodnota je uvedena jako první. Tato posloupnost má pak obecný tvar: n, x_1, x_2, ….., x_i, ……, x_n,

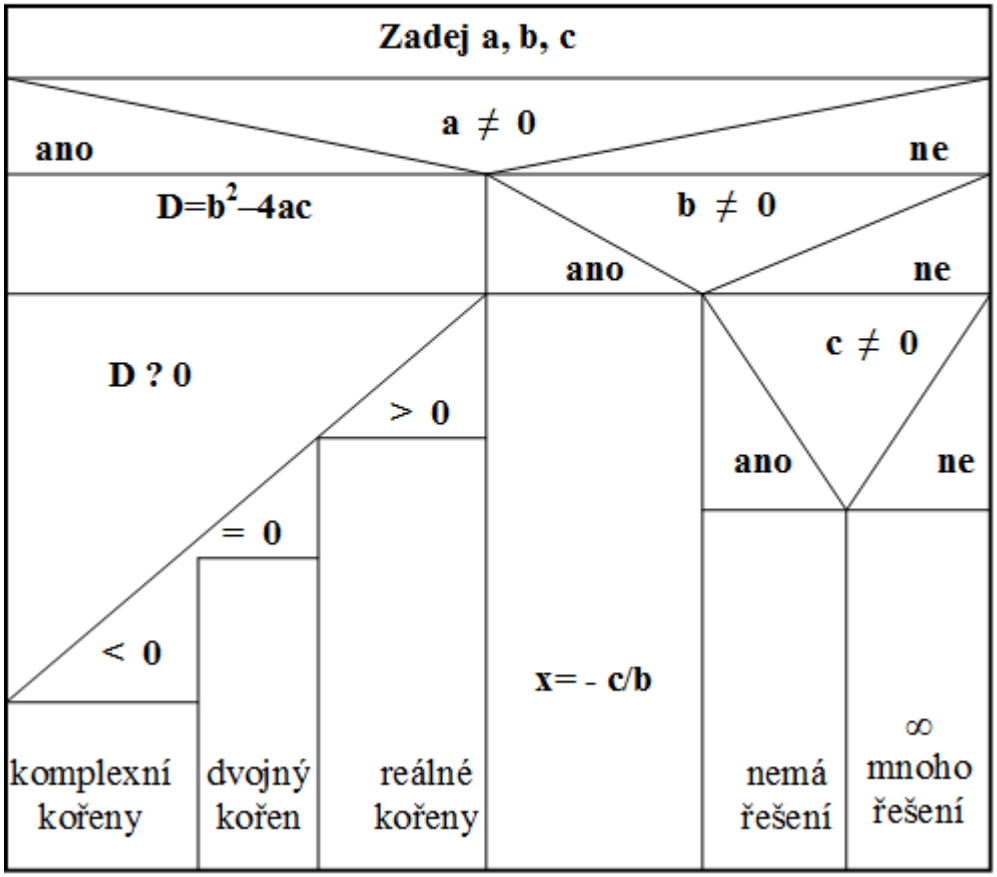


Struktogramy

Nassi Shneiderman diagrams [on line, cit. 2018-09-17]



Struktogramy - příklad



Grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů:
UML (Unified Modeling Language)
[on line, cit. 2020-09-20].
Bude prezentován v předmětu IUS.

Rozhodovací tabulky

- rozhodovací tabulky byly definovány zejména pro algoritmizaci úloh se složitým rozhodováním v oblasti zpracování hromadných dat
- jsou nástrojem k vyjádření komplexní logiky rozhodování relativně jednoduchým způsobem
- v rozhodovacích tabulkách se postupně znázorňují variantní série činností, které se mají provést při různých kombinacích výchozích podmínek

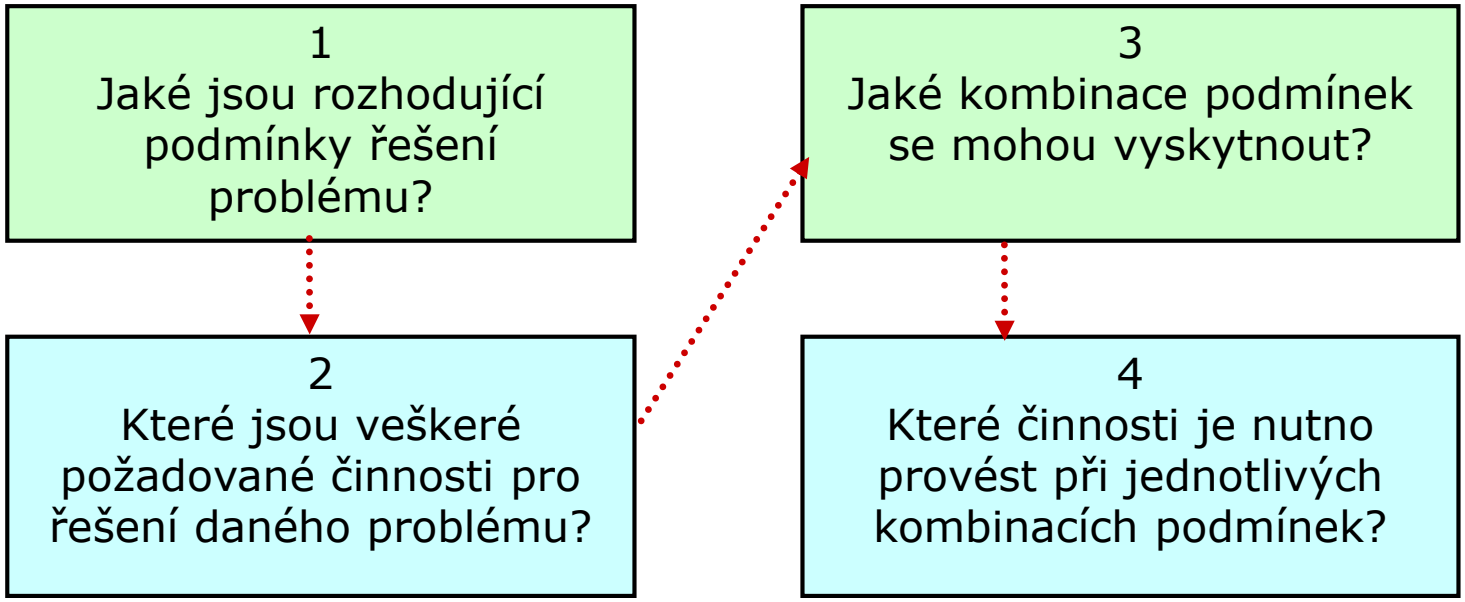
Rozhodovací tabulky

Základní tvar rozhodovací tabulky:

Název tabulky	**Záhlaví pravidel**
1 Seznam podmínek	3 Kombinace podmínek
2 Seznam činností	4 Kombinace činností

Rozhodovací tabulky

Postup vyplňování rozhodovací tabulky:



Rozhodovací tabulky - příklad

Příklad: Rozhodovací tabulka pro rozhodování při nečinnosti tiskárny.

Podmínky	Tiskárna netiskne	A	A	A	A	N	N	N	N
	Svítí červená kontrolka	A	A	N	N	A	A	N	N
	Tiskárna nebyla rozpoznána	A	N	A	N	A	N	A	N
Akce	Ověř síťový kabel			X					
	Ověř tiskárna-PC kabel	X		X					
	Ověř driver	X		X		X		X	
	Ověř-doplň toner	X	X			X	X		
	Zjisti zda není zaseknutý papír		X		X				

Programovací jazyky

- Syntaxe a sémantika programovacích jazyků.
- Syntaktické diagramy.
- BNF (Backus Naurova Forma), EBNF.

Programovací jazyky

[on line, cit. 2020-09-22]

- Přehled aktuálnosti programovacích jazyků
- seznam programovacích jazyků
- popularita jazyků

Syntaxe a sémantika programovacích jazyků

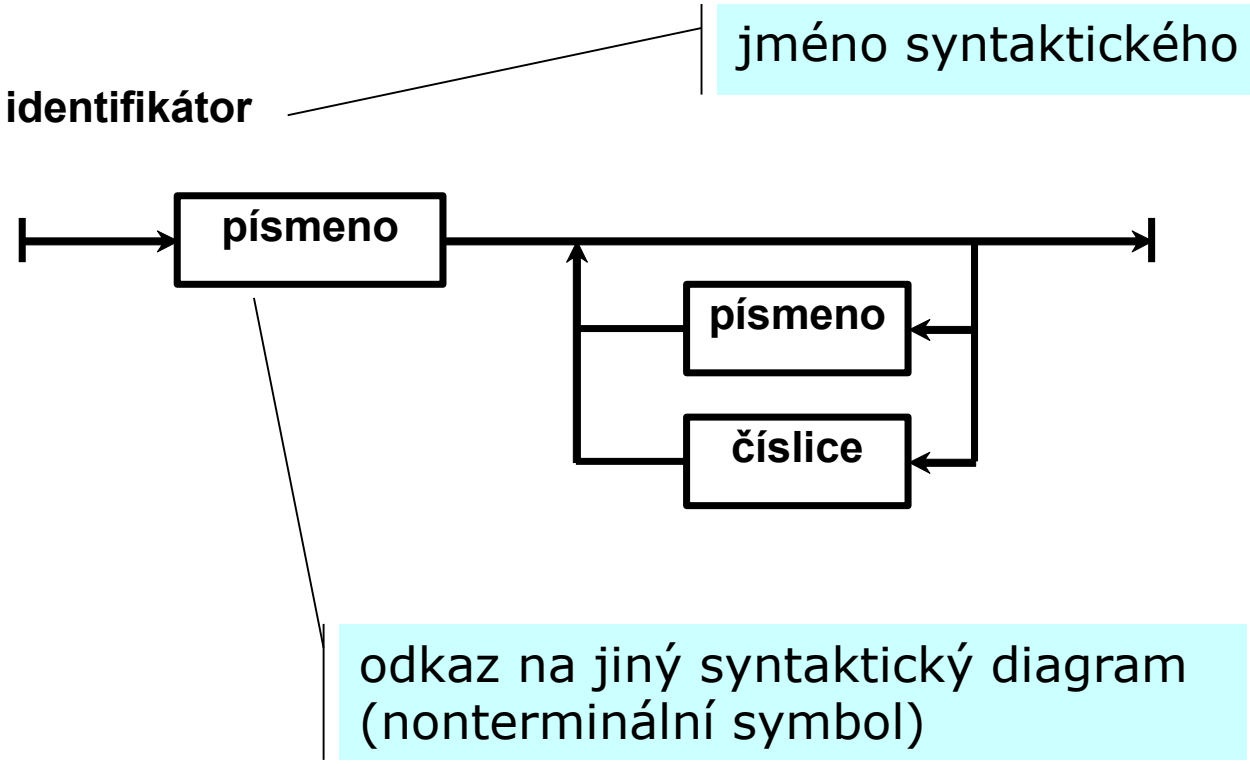
- **syntaxe** - soubor pravidel udávající přípustné konstrukce programů
  - popisuje formální strukturu programu
  - definuje klíčová slova, identifikátory, čísla a další programové entity a určuje způsob, jak je lze kombinovat
  - na základě syntaktických pravidel lze posoudit, zda určitý text je či není korektním zápisem programu v daném jazyce
- **sémantika** určuje logický význam jednotlivých výrazů jazyka
  - ze sémantiky plyne, jaký má daná konstrukce význam

Syntaxe a sémantika programovacích jazyků

- Pro popis syntaxe programovacích jazyků lze použít :
  - syntaktické diagramy (grafický popis)
  - Backus-Naurova forma (BNF), EBNF

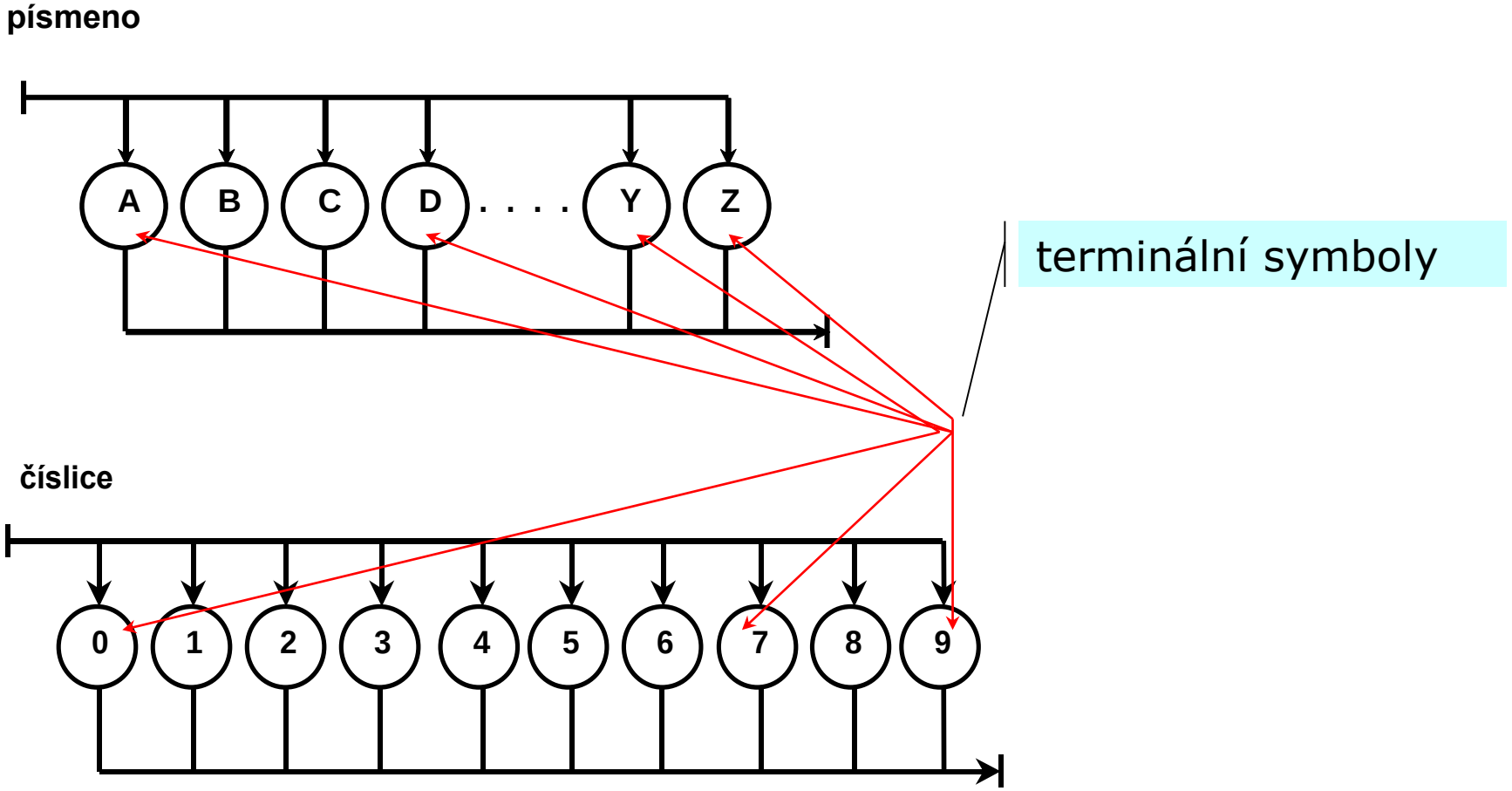
Syntaktické diagramy

Příklad 1: syntaktický diagram identifikátoru



Syntaktické diagramy

Příklad 1: syntaktický diagram identifikátoru



Syntaktické diagramy

- **terminální symbol** - dále se nerozgenerovává (znak, klíčové slovo...), je zapsán v kroužku nebo oválu
- **nonterminální symbol** - rozgenerovává se podle odkazujícího syntaktického diagramu, je zapsán v obdélníku

Backus-Naurova forma (BNF)

- BNF má stejnou vyjadřovací schopnost (mocnost) jako grafy syntaxe
- obě struktury jsou ekvivalentní a vzájemně převoditelné
- často se používá pro popis parametrů programů ovládaných z příkazového řádku
- existuje mnoho mutací

Backus-Naurova forma (BNF)

Základem jsou dva typy symbolů:

- **terminální** symboly (terminály)
  - jsou to prvky základní abecedy jazyka (klíčová slova, operátory, závorky, …)
- **nonterminální** symboly (nonterminály)
  - jsou dále rozgenerovávány
  - nahrazují podrobněji popsané elementy

Backus-Naurova forma (BNF)

Součástí BNF jsou metajazykové symboly:

- **<>** úhlové závorky ohraničují nonterminály
- **=** definiční znak, nalevo od něj stojí nonterminál, napravo jeho definice

  (::=, —>>)
- **|** alternativní znak - odděluje seznam variant, varianta může být i prázdná (někdy se zapisuje znakem "e").

Pozor na kolize metajazykových symbolů a terminálů popisovaného jazyka! (Lze řešit např. pomocí uvozovek)

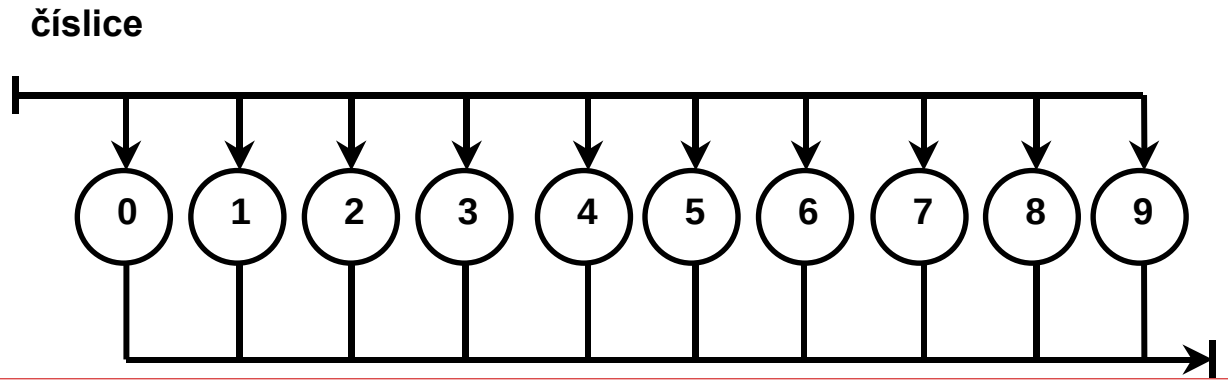
Backus-Naurova forma (BNF)

Příklad 1 : syntaxe číslice

BNF:

<číslice> = 0|1|2|3|4|5|6|7|8|9

Syntaktický diagram:



Backus-Naurova forma (BNF)

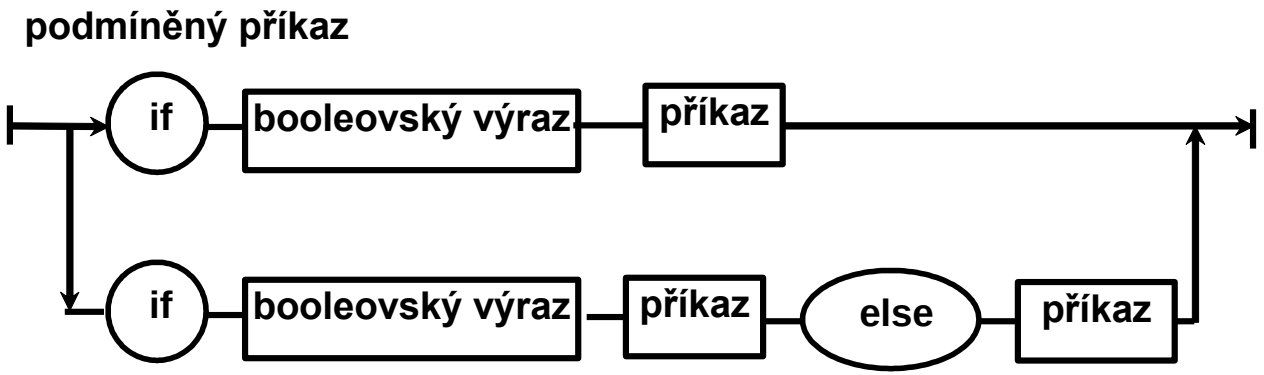
Příklad 2 : syntaxe podmíněného příkazu

BNF:

<podmíněný příkaz> = if <booleovský výraz> <příkaz> |

if <booleovský výraz> <příkaz> else <příkaz>

Syntaktický diagram:



EBNF - rozšíření

- rozšíření zjednodušují zápis, ale nezvyšují vyjadřovací schopnosti formy
  - { } řetězec terminálních a nonterminálních symbolů uzavřený ve složených závorkách je opakován nula až n-krát
  - [ ] řetězec terminálních a nonterminálních symbolů uzavřený v hranatých závorkách je nepovinný
  - ( ) seskupování konstrukcí

EBNF - rozšíření

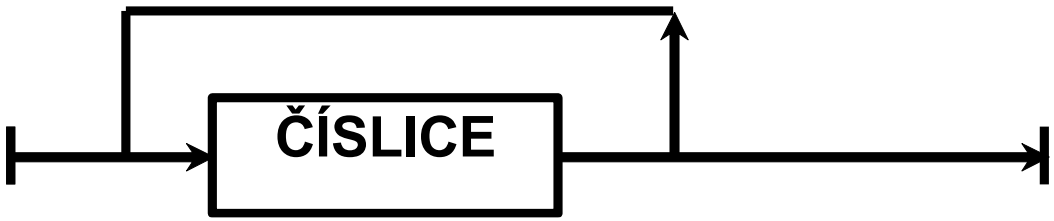
Příklad 3 : celé číslo

EBNF:

CELÉ ČÍSLO = ČÍSLICE, { ČÍSLICE }

Syntaktický diagram:

CELÉ ČÍSLO



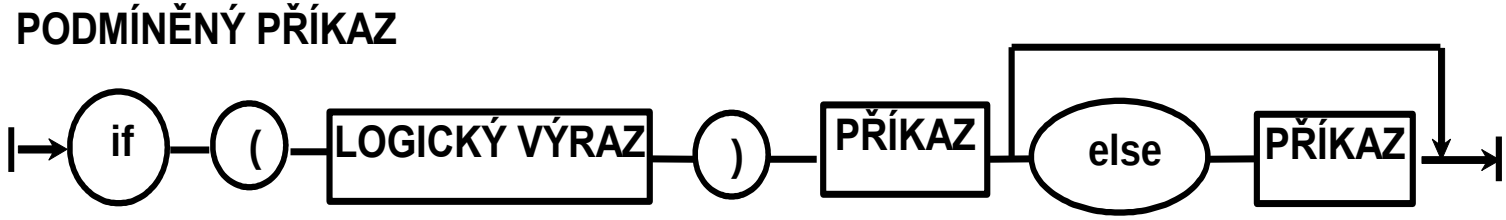
EBNF - rozšíření

Příklad 4 : syntaxe podmíněného příkazu

EBNF:

PODMÍNĚNÝ PŘÍKAZ = **if,** (LOGICKÝ VÝRAZ), PŘÍKAZ, [**else,** PŘÍKAZ]

Syntaktický diagram:



Sémantika

Příklad 5 : sémantika podmíněného příkazu

Po vyhodnocení logického výrazu (musí být v závorkách) se v případě jeho nenulové hodnoty (true) provede příkaz za logickým výrazem. Po jeho provedení pokračuje program za tímto příkazem. V případě nulového výsledku (false) výrazu se řízení programu předá bezprostředně za podmíněný příkaz. Jinak řečeno se příkaz za logickým výrazem přeskočí.
Příkaz **if** můžeme použít i s variantou **else**. Sémantika takového příkazu **if-else** je v první části totožná se samotným příkazem **if**. Je-li výslednou hodnotou logického výrazu hodnota jedna/nenulová hodnota (true), provede se příkaz za logickým výrazem. V opačném případě, kdy výsledkem je nula, se provede příkaz za **else**. V obou případech se řízení programu po provedení prvního, respektive druhého příkazu předá za celý podmíněný příkaz.

BNF, EBNF - odkazy

- syntaktický diagram i BNF/EBNF jsou prostředky ekvivalentní gramatice
- BNF (Backusova Naurova forma))

[on line, cit. 2019-09-02]

- ISO/IEC 14977:1996(E)

  EBNF Syntax Notation,

  final draft version (SC22/N2249)

  Syntax (XBNF)

[on line, cit. 2019-09-17]

Další způsoby popisu programovacího jazyka

- **tutoriál**
  - základní průvodce daným programovacím jazykem
  - dává základní představu o hlavních rysech a konstrukcích jazyka
  - problematika je typicky vysvětlována na příkladech
  - syntaxe a sémantika jazyka je uváděna postupně, podle potřeby
  - slouží k základnímu seznámení s daným programovacím jazykem

Další způsoby popisu programovacího jazyka

- **uživatelský manuál**
  - manuál, popisující práci s příslušným překladačem z uživatelského hlediska
  - důraz je zejména kladen na technické otázky jako jsou:
    - instalace produktu
    - jeho spouštění
    - práce v menu atd.

Další způsoby popisu programovacího jazyka

- **referenční manuál**
  - manuál, popisující vyčerpávajícím způsobem syntaxi a sémantiku programovacího jazyka
  - je tradičně tříděn abecedně, podle syntaxe jazyka
  - jednotlivé příkazy jazyka jsou popsány přirozeným jazykem (tj. většinou anglicky)
  - slouží zejména programátorům pro rychlé vyhledání podrobností o jednotlivých příkazech

Další způsoby popisu programovacího jazyka

- **formální definice**
  - precizní definice jazyka, obvykle formou závazné normy určená pro profesionály v daném oboru
  - používá formalismus BNF/EBNF nebo syntaktických diagramů

Úvod do algoritmizace

Úkoly k procvičení

Zpracování posloupností hodnot.
Vytvořte vývojový diagram pro:

1. součet n hodnot.
2. počet záporných, nulových a kladných hodnot z n hodnot.
3. aritmetický průměr kladných hodnot z n hodnot.
4. největší hodnota z n hodnot.
5. nejmenší a největší hodnota z n hodnot.
6. 2 největší hodnoty z n hodnot.
7. největší hodnota a počet jejích výskytů z n hodnot.
8. nejmenší hodnota a pořadí jejího prvního výskytu z n hodnot.
9. jsou zadány známky 1-5, koncová hodnota je 0. Určete počet jedniček, dvojek, trojek čtyřek, pětek.
10. jsou dána kladná celá čísla zakončená číslem nula. Určete nejmenší číslo.

Kontrolní otázky

1. Vyjmenujte etapy programátorské práce a stručně je charakterizujte.
2. Co je algoritmus?
3. Vysvětlete vlastnosti algoritmů.
4. Co jsou vývojové diagramy a k čemu slouží?
5. Vysvětlete co je syntaxe a sémantika programovacích jazyků.
6. Co je BNF, EBNF? Metajazykové symboly.
7. Co je syntaktický diagram, z čeho se skládá, k čemu slouží?