Algoritmy, datové a procesní struktury

Algoritmus

- Není výsadou IT
- Přesný návod či postup, kterým lze vyřešit danou úlohu (typ úlohy).
- V IT abstraktně popsaný postup řešení problému.
- Na algoritmy jsou vyšší požadavky než na plány.

Atributy algoritmů

- Rezultativnost – vede k vytčenému cíli
- Konečnost – konečný počet kroků algoritmu
- Determinovanost – v každém okamžiku je jasné, jak postupovat
- Elementárnost – užití základních procedur a funkcí kompatibilních s cílovým zařízením
- (Efektivnost) – vyřešení úkolu s minimálními použitými prostředky a v nejkratším čase
- (Hromadnost, univerzalita) – opakovatelné použití (parametrizace)

Příklad

- Nákup v e-shopu
- Cesta na dovolenou
- Návod k mobilu

Datové a procesní struktury

- Stavební kameny pro tvorbu algoritmů
- Nezávislé na prostředí a implementaci
  - použitelné obecně pro většinu cílových systémů
- Vázané na konkrétní technické možnosti cílového systému
  - hlavně interakce konkrétního systému s okolím

Obecné procesní kroky

- Výkonný (operační) krok
  - provedení elementární akce, koncový systém jí rozumí
  - nelze nebo není nutné jej dále dělit a skládat z dílčích kroků
  - příklad: jdi_vpřed, vypočti c=a+b

- Příkaz skoku v algoritmu
  - skok vpřed – vypuštění části algoritmu
  - skok vzad – opakované provedení části algoritmu
  - nemusí být implementován v konkrétním jazyce

- Větvení (rozhodovací krok)
  - výběr dalšího postupu podle vyhodnocení podmínky
  - pro logické podmínky 2 cesty, ale může být i více
  - podmínka musí být vyhodnotitelná
  - podmínka může a nemusí vycházet z předchozích kroků algoritmu
  - 1+1, 1=1, x^2, 2x=3y

Další procesní kroky

- Cyklus
  - vhodná kombinace obecných kroků větvení a skoku
  - blok kroků prováděný opakovaně
  - předem známý nebo neznámý počet opakování
  - příklad: součet čísel od 1 do 100

- Vstup a výstup - interakce s okolím
  - kroky zajišťující nutné vstupní údaje a poskytující informace o činnosti algoritmu a jeho výsledcích

Ukládání dat

- Paměťový prostor pro data, se kterými pracujeme
  - bere v úvahu typ ukládaných dat
  - obvykle k dispozici po dobu vykonávání algoritmu
- Proměnná
  - paměťové místo určené k ukládání dat v průběhu vykonávání algoritmu
  - obsah proměnné se může měnit
- Konstanta
  - specifický typ proměnné
  - symbol zastupující konkrétní a po dobu vykonávání algoritmu neměnnou hodnotu (π)
- Označení místa v paměti identifikátory
  - x, muj_vek, y12, …
  - musí začínat písmenem, ideální bez diakritiky, mezer a speciálních znaků

Dostupnost dat

- Proměnné a konstanty nemusí být dostupné všude v algoritmu
- Různý rozsah platnosti
  - globální – lze s nimi pracovat všude v algoritmu
  - lokální – jsou dostupné jen v části algoritmu (v samostatné a ohraničené části)
- Podle okamžiku vzniku
  - statické – jsou známé již při přípravě algoritmu
  - dynamické – jsou vytvářeny během činnosti algoritmu (např. položky v adresáři)

Jednoduché datové typy

- Číselné (integer, real, byte)
  - datum a čas
- Znakové – kódování znaků
  - ASCII, UTF8
- Textové – sekvence znaků
- Boolean – logické ano/ne

Datové struktury

- Jednoduché – některého ze základních typů
  - číslo (teplota), text (moje jméno), ano/ne (je den)
- Složené – kolekce údajů
  - s různým významem a často různého typu (soubor údajů o jedné osobě, objekt, záznam)
  - se stejným typem a významem (seznam, pole, kalendář, předměty ve škole)
  - uspořádané a neuspořádané (seznam vs. množina)

Strukturované datové typy

- Obsahují prvky elementárních nebo strukturovaných typů
  - rekurzivní definice
- Kolekce
  - elementy stejného typu
  - statické – pole s daným počtem prvků
  - dynamické – seznamy (jednosměrný, obousměrný, podle typu obsluhy FIFO, LIFO), mapy, stromy (binární, vyvážené)
- Soubory
  - binární x textový
  - sekvenční a náhodný přístup

Prostředky k vyjádření algoritmů

- Přirozený jazyk
- Jazyk matematiky a logiky
  - rozhodovací tabulky
- Grafické prostředky
  - vývojové diagramy – obecný vizuální jazyk
- Pseudojazyky
  - specifický textový popis
  - liší se od programovacích jazyků volnějšími pravidly syntaxe – nejde je přímo implementovat
- Programovací jazyky

Příklad: Hledání NSD

Úloha:

Najděte největšího společného dělitele (NSD) čísel *6* a *15*

Řešení:

Popišme **postup** tak, aby byl použitelný pro dvě **libovolná** přirozená čísla (nejen pro ***6*** a ***15***):

- označme zadaná čísla ***x*** a ***y*** a menší z nich ***d***
- není-li ***d*** společným dělitelem ***x*** a ***y***, pak zmenšíme ***d*** o ***1***, test opakujeme a skončíme, až ***d*** bude společným dělitelem ***x*** a ***y***

Poznámka:

Význam symbolů ***x***, ***y*** a ***d*** použitých v postupu řešení (algoritmu):

- jsou to **proměnné** (paměťová místa), ve kterých je uložena nějaká **hodnota**, která se může v průběhu výpočtu měnit

Algoritmus hledání NSD - provádění

Provádění algoritmu hledání NSD čísel 6 a 15 po jednotlivých dílčích operacích:

1. Zahájení provádění, vstup a označení (přiřazení) čísel *x*,*y*,*d*
2. Testování dělitelnosti čísel *x* a *y* číslem *d*
3. Zmenšení čísla *d* o *1*
4. Ukončení provádění a výstup výsledku

Krok č.	Operace č.	*x*	*y*	*d*	Vyhodnocení operace a stanovení další operace
1.	1	6	15	6	**Vstup** a **přiřazení** čísel, pokračuj operací 2
2.	2	6	15	6	***d*** není dělitelem ***y***, pokračuj operací 3
3.	3	6	15	5	Snížení hodnoty ***d***, opakuj operaci 2
4.	2	6	15	5	***d*** není dělitelem ***x***, opakuj operaci 3
5.	3	6	15	4	Snížení hodnoty ***d***, opakuj operaci 2
6.	2	6	15	4	***d*** není dělitelem ***x*** ani ***y***, opakuj operaci 3
7.	3	6	15	3	Snížení hodnoty ***d***, opakuj operaci 2
8.	2	6	15	3	***d*** je dělitelem ***x*** i ***y***, pokračuj operaci 4
9.	4	6	15	3	**Výsledek** je číslo **3** a **konec** provádění

Algoritmus - zobecnění a slovní zápis

Úloha: najděte největšího společného dělitele (nsd) dvou přirozených čísel

Přesnější popis:

Vstup: přirozená čísla ***x*** a ***y***

Výstup: ***nsd(x,y)***

Postup:

1. Je-li ***x<y***, pak ***d*** má hodnotu ***x***, jinak ***d*** má hodnotu ***y***
2. Opakuj krok 3., pokud ***d*** není dělitelem ***x*** ***nebo*** *d* není dělitelem ***y***
3. Zmenši ***d*** o ***1***
4. Výsledkem je hodnota ***d***

Sestavili jsme **algoritmus** pro výpočet **největšího společného dělitele** dvou přirozených čísel.

Algoritmy - pseudojazyk

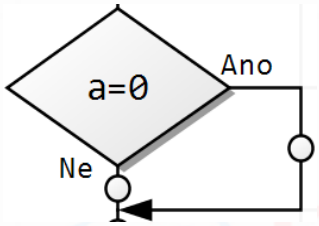
Zápis algoritmu v pseudojazyce (připouští i slovní formulace operací):

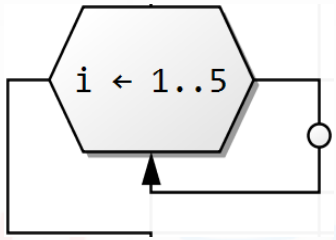
```
nsd(x,y):
if x<y then d:=x else d:=y;
while d "není dělitelem" x or d "není
 dělitelem" y do
    d:=d-1;
nsd:=d;
```

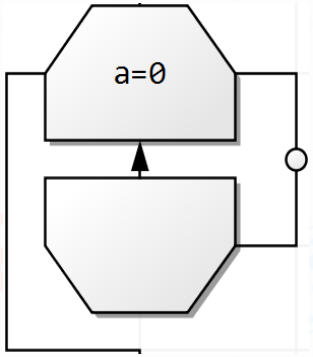
Zápis algoritmu v programovacím jazyku:

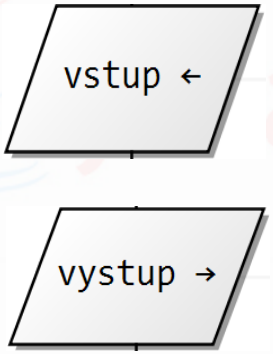
```
int nsd(int x, int y)
{
 int d;
 if (x<y) d=x;
 else d=y;                          // operace % je tzv. zbytek po dělení
 while (x%d!=0 | y%d!=0)
   d--;
 return d;
}
```

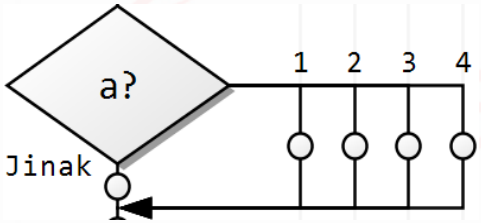
Vývojové diagramy

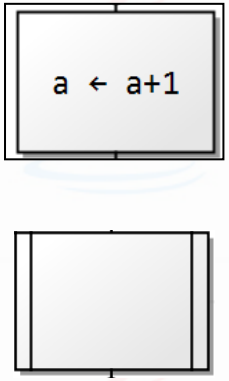


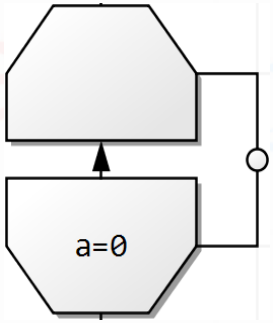






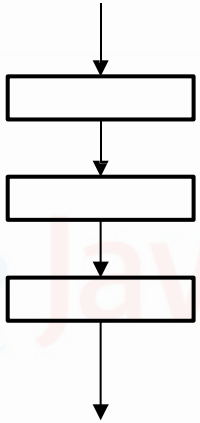




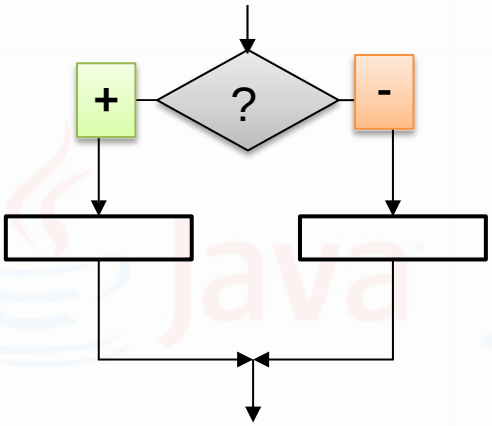


Řídící struktury

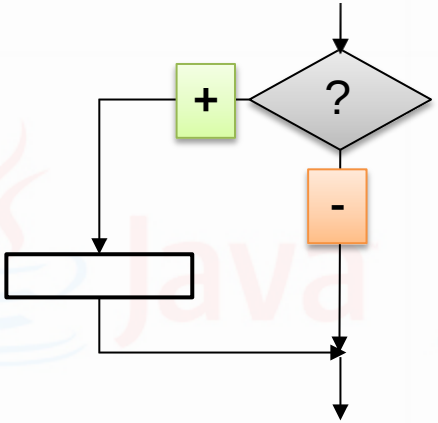
sekvence



if - else

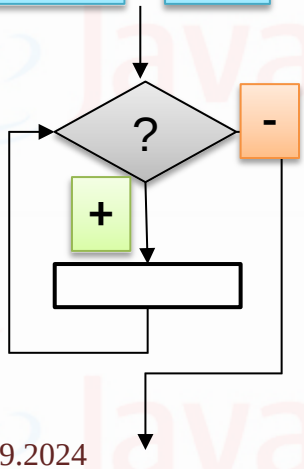


if

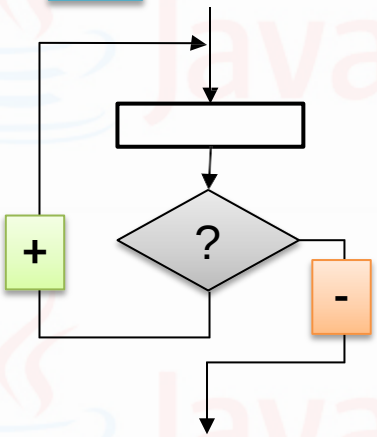


while

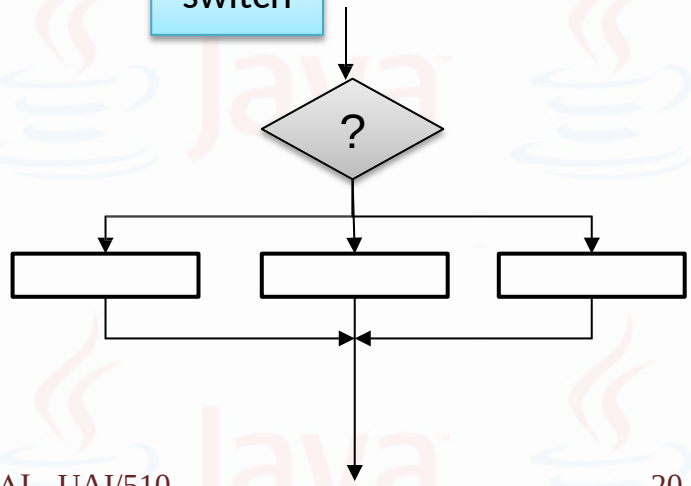
for



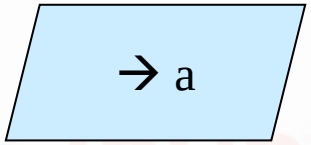
do

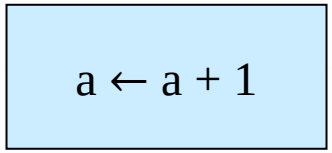


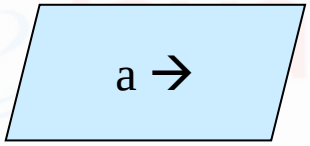
switch



Výkonný krok a vstup / výstup



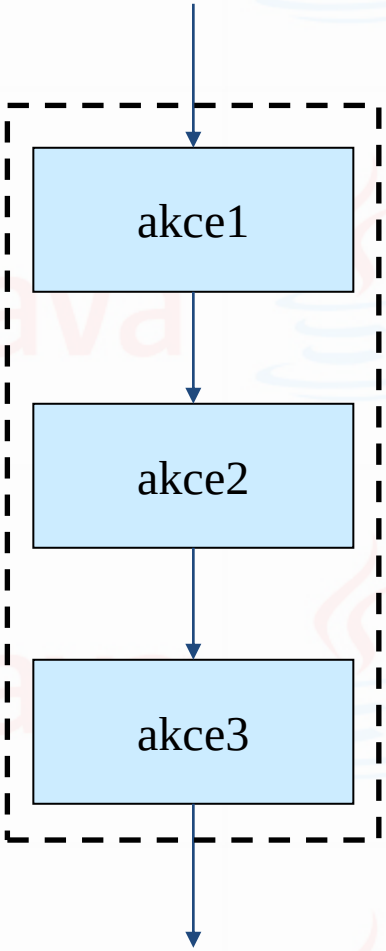


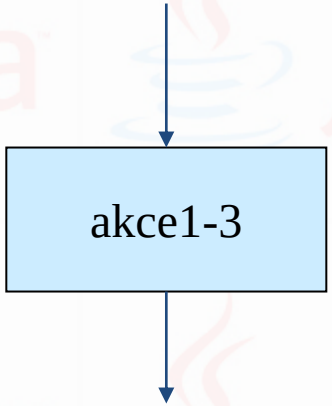


Posloupnost tvoří sled akcí (elementárních, funkcí, procedur) navazujících logicky na sebe

Pozn.: vstup od výstupu odlišují šipky ukazující směr toku dat.

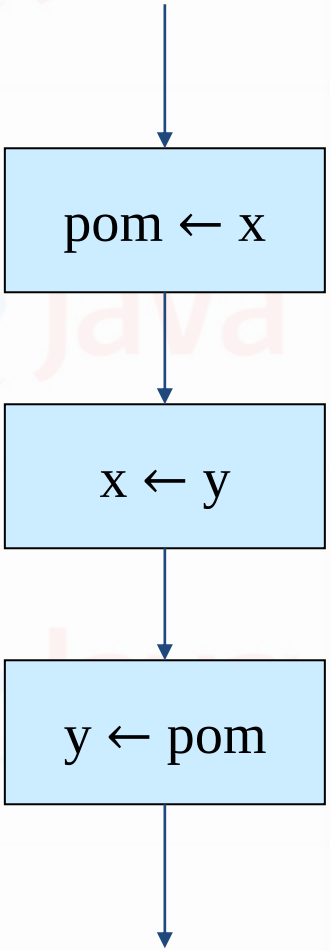
Sekvence



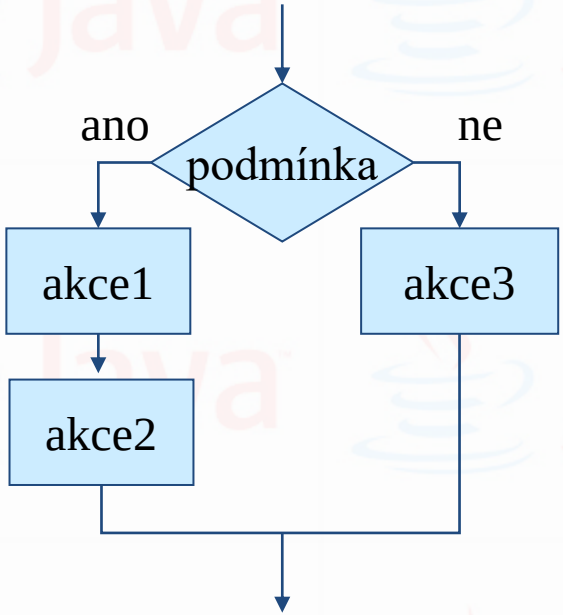


Blok

Výměna obsahu proměnných *x* a *y*

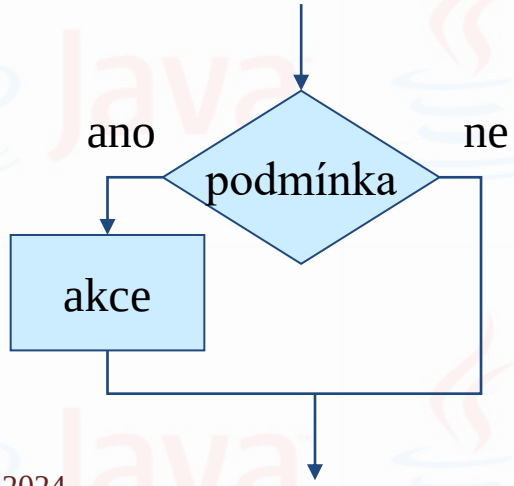


Větvení



Význam:

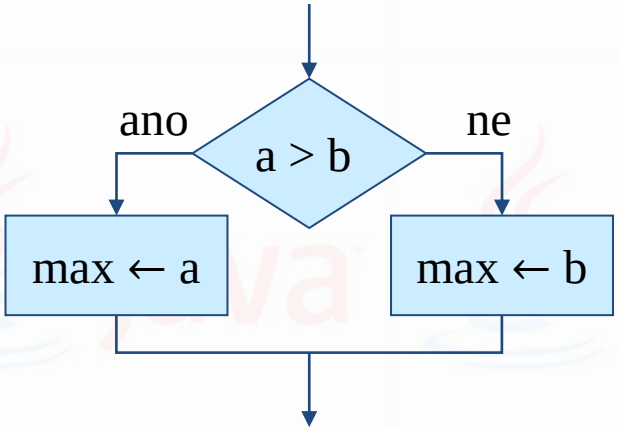
jestliže platí *podmínka*, pak proveď *akce1 a akce2*, jinak proveď *akce3*.



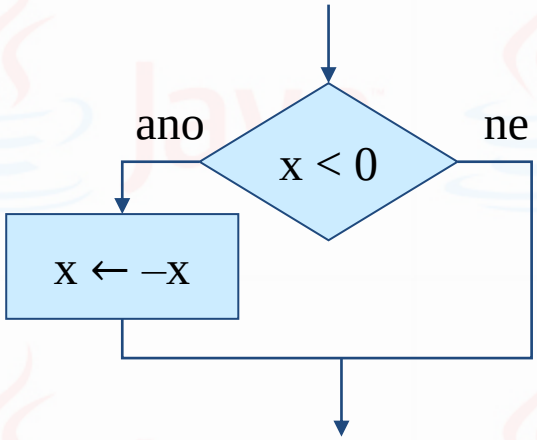
Význam:

jestliže platí *Podmínka*, pak proveď *akce*.

Příklady větvení

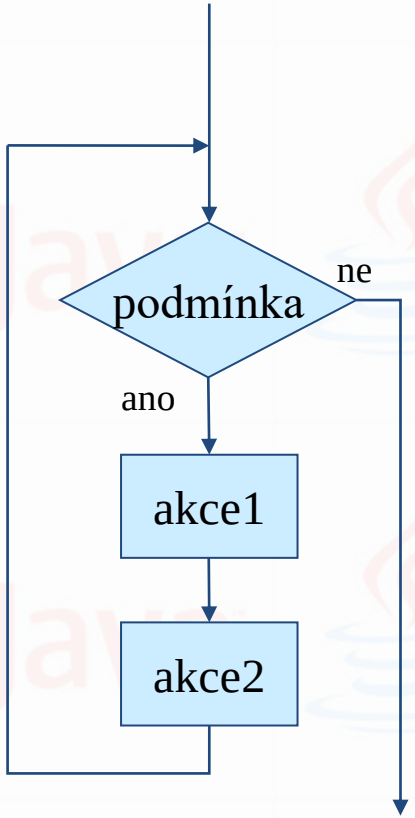


Určení maxima z čísel *a* a *b*



Náhrada hodnoty proměnné *x* její absolutní hodnotou

Iterace (cyklus) s podmínkou na začátku

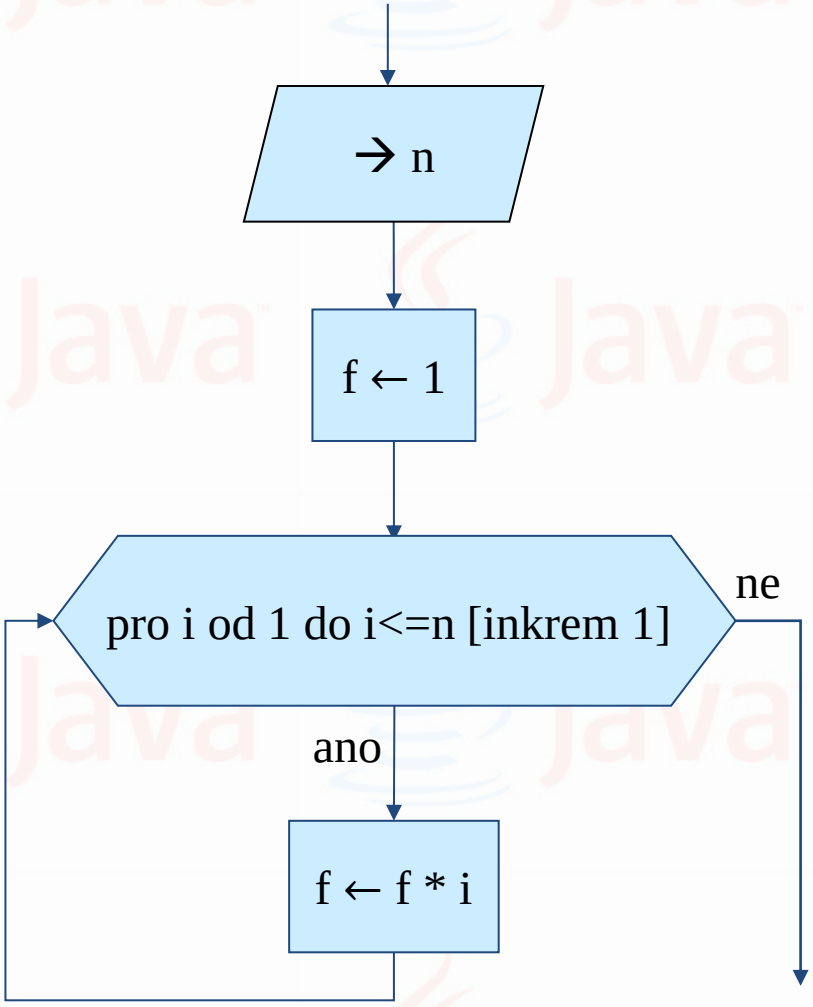


Význam:

pokud platí *podmínka*,
prováděj *akce1*, *akce2*.

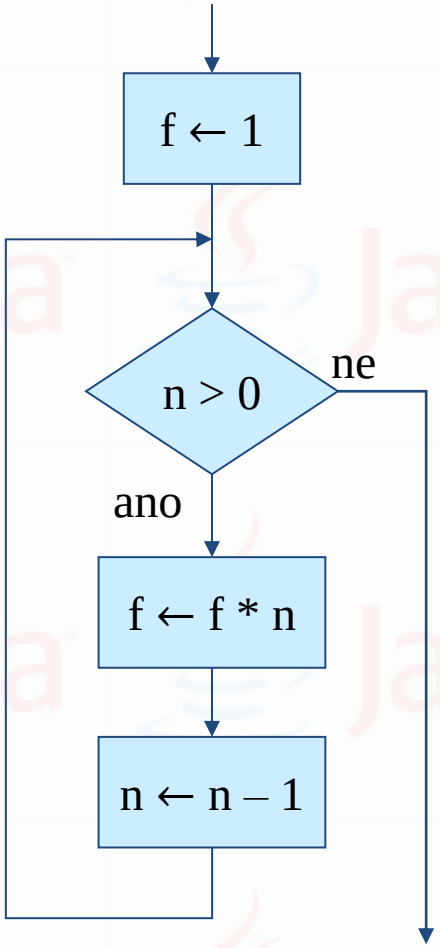
- Pozn.1: akce se nemusí provést ani jednou (když podmínka není splněna hned napoprvé).
- Pozn.2: akce musí mít vliv na vyhodnocení podmínky, aby iterace skončila v konečném počtu kroků.
- Pozn.3: Umožňuje i realizaci iterace s daným počtem průchodů – cyklus FOR.

Iterace s pevným počtem průchodů

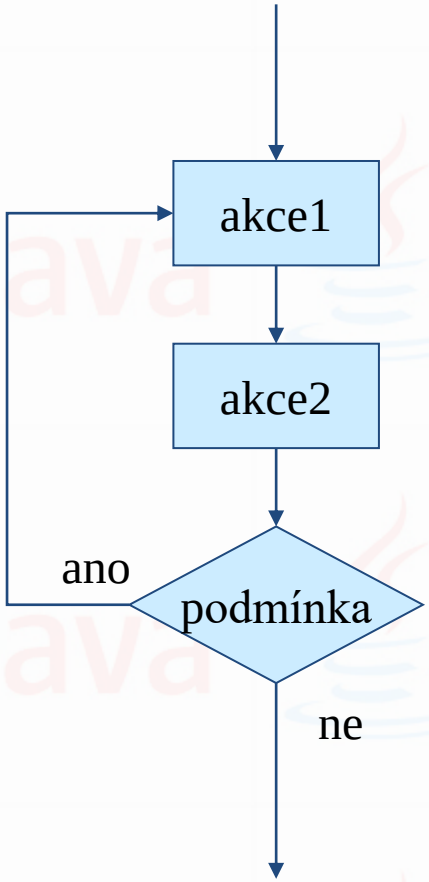


Výpočet faktoriálu *f* přirozeného čísla *n*

Výpočet faktoriálu *f* přirozeného čísla *n*



Iterace (cyklus) s podmínkou na konci

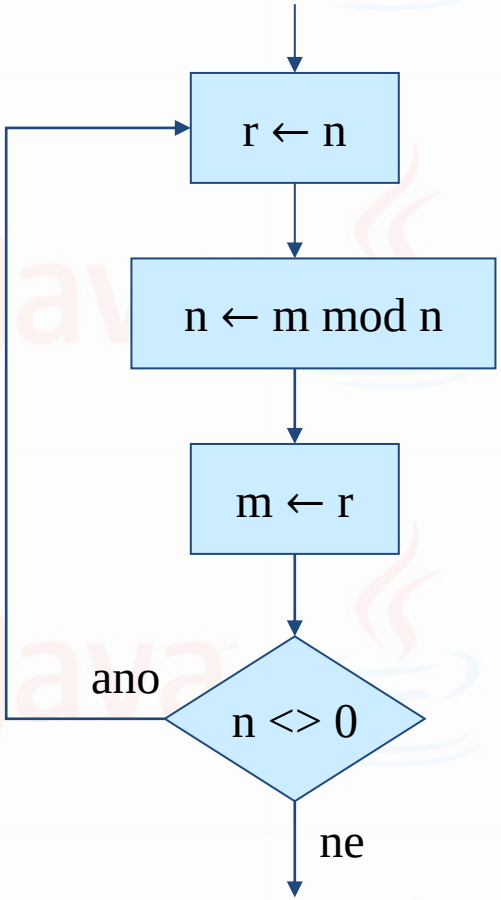


Význam:

opakuj *akce1* a *akce2*
dokud platí *podmínka*

- Pozn.: Akce se provedou vždy alespoň jednou.

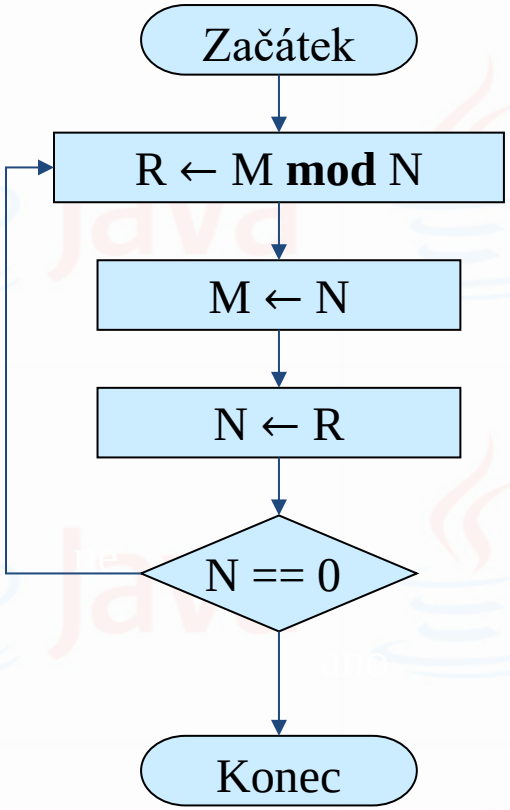
Příklad



Euklidovský výpočet největšího společného dělitele dvou celých kladných čísel *m* a *n*

- Pozn1: m > n, n ≠ 0
- Pozn2: Výsledek je v m.

Největší společný dělitel

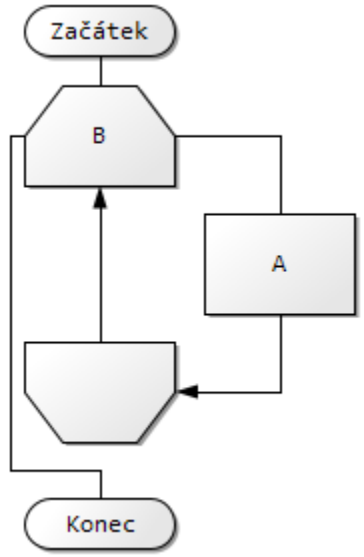


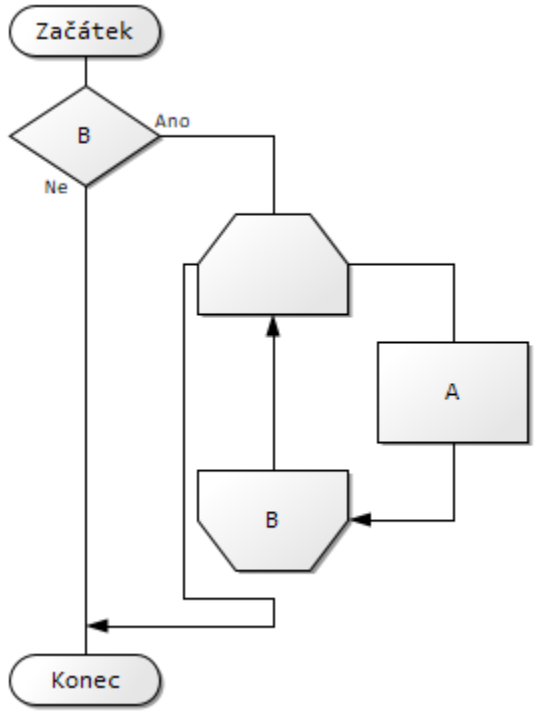
Operace **mod** dává zbytek po celočíselném dělení

Hodnota největšího společného dělitele se nachází v proměnné M

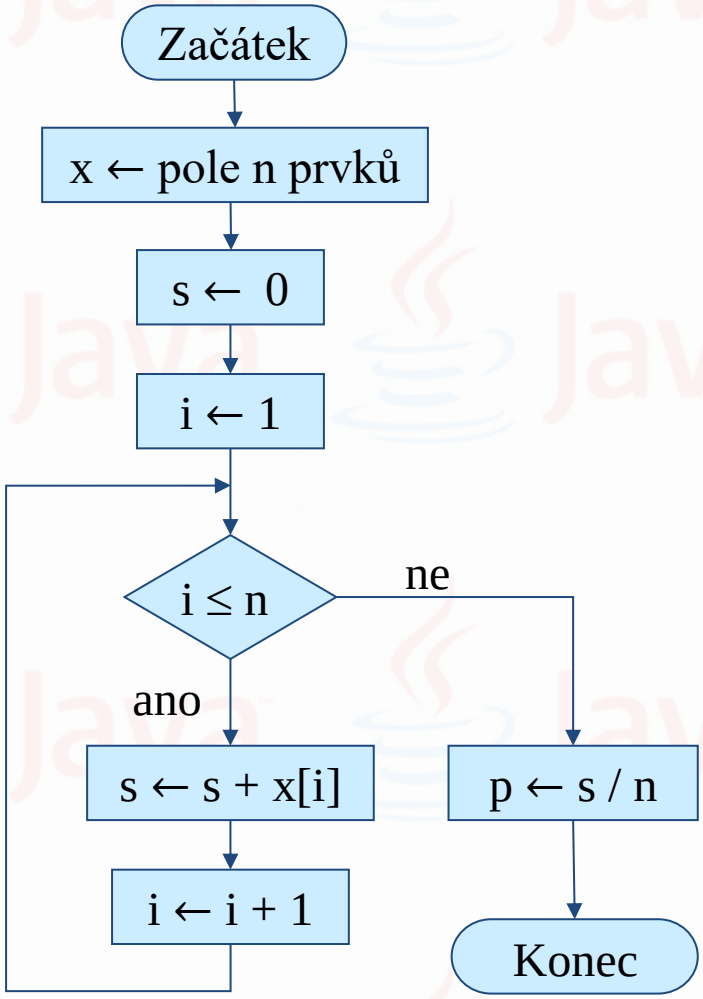
Vztahy mezi cykly

- Cykly s podmínkou na začátku lze transformovat do cyklů s podmínkou na konci a naopak





Příklad práce s polem



Aritmetický průměr p čísel $x_1, \ldots, x_n$ v poli x

Pozn.: Pole jsou v programovacích jazycích indexována od 0.

Prostředky pro práci s algoritmy

- Buď pouze návrh algoritmu nebo i jeho ověření, implementace a užití

- PSDiagram - http://www.psdiagram.cz/
- Snap! - http://snap.berkeley.edu/
- AppInventor - http://appinventor.mit.edu/
- Kodu - https://www.kodugamelab.com/
- …

Algoritmus - shrnutí

Algoritmus

- postup při řešení určité třídy úloh, který je tvořen souborem jednoznačně definovaných kroků (tj. příkazů) a zaručuje, že pro každou přípustnou kombinaci vstupních dat se po provedení konečného počtu kroků dospěje k požadovaným výsledkům

Vlastnosti algoritmu:

- hromadnost
  měnitelná vstupní data
- determinovanost
  každý krok je jednoznačně definován
- konečnost a resultativnost
  pro přípustná vstupní data se po provedení konečného počtu kroků dojde k požadovaným výsledkům

Algoritmus = syntetický model postupu řešení obecných úloh

Prostředky pro zápis algoritmu:

- přirozený jazyk, vývojové diagramy, struktogramy, pseudojazyk, programovací jazyk