# 1. Algoritmus a vývojový diagram

## Teoretická část

**Popište a vysvětlete, co je to algoritmus, jaké musí splňovat vlastnosti a jaké má druhy?**  
**Popište znázornění algoritmu vývojovým diagramem.**

## Praktická část

1. **Nakreslete a obhajte vývojový diagram „nalezení nejvyššího čísla z 3“ pro čísla A, B, C.**
2. **Nakreslete a popište u jednoduchého algoritmu funkci podmínky a cyklu se zpětnou vazbou.**
3. **Popište algoritmus obecného přihlášení s kontrolou špatných zadání.**

## Doplňující otázky

1. **Popište a vysvětlete funkci heuristického algoritmu.**
2. **Nakreslete a popište elementární prvky používané ve vývojovém diagramu.**
3. **Popište rozdílnost algoritmů pravděpodobnostních a genetických.**

# Algoritmus

**Algoritmus** je přesný **návod či postup**, kterým lze vyřešit daný typ úlohy. Pojem algoritmu se nejčastěji objevuje při [programování](https://cs.wikipedia.org/wiki/Programov%C3%A1n%C3%AD), kdy se jím myslí teoretický princip řešení problému (oproti přesnému zápisu v konkrétním [programovacím jazyce](https://cs.wikipedia.org/wiki/Programovac%C3%AD_jazyk)).

Obecně se ale algoritmus může objevit v jakémkoli jiném vědeckém odvětví. Jako jistý druh algoritmu se může chápat i např. kuchařský recept. V užším smyslu se slovem algoritmus rozumí takové postupy, které splňují některé silnější požadavky.

## Vlastnosti:

**Výstup (resultativnost)**

*Algoritmus má alespoň jeden výstup, veličinu, která je v požadovaném vztahu k zadaným vstupům, a tím tvoří odpověď na problém, který algoritmus řeší (algoritmus vede od zpracování hodnot k výstupu)*

**Obecnost (hromadnost, masovost, univerzálnost)**

*Algoritmus neřeší jeden konkrétní problém (např. „jak spočítat 3×7“), ale obecnou třídu obdobných problémů (např. „jak spočítat součin dvou celých čísel“), má širokou množinu možných vstupů.*

**Determinovanost**

*Každý krok algoritmu musí být jednoznačně a přesně definován; v každé situaci musí být naprosto zřejmé, co a jak se má provést, jak má provádění algoritmu pokračovat, takže pro stejné vstupy dostaneme pokaždé stejné výsledky. Pro zápis algoritmů byly navrženy*[*programovací jazyky*](https://cs.wikipedia.org/wiki/Programovac%C3%AD_jazyk)*, ve kterých má každý příkaz jasně definovaný význam. Některé algoritmy ale determinované nejsou, pravděpodobnostní algoritmy v sobě mají zahrnutu náhodu.*

**Elementárnost**

*Algoritmus se skládá z konečného počtu jednoduchých (elementárních) kroků.*

**Konečnost (finitonost)**

*Každý algoritmus musí skončit v konečném počtu kroků. Tento počet kroků může být libovolně velký (podle rozsahu a hodnot vstupních údajů), ale pro každý jednotlivý vstup musí být konečný.*

## Druhy algoritmů

Algoritmy můžeme klasifikovat různými způsoby. Mezi důležité druhy algoritmů patří:

* [**Rekurzivní**](https://cs.wikipedia.org/wiki/Rekurze)**algoritmy**, které využívají (volají) samy sebe.
* [**Pravděpodobnostní algoritmy**](https://cs.wikipedia.org/wiki/Pravd%C4%9Bpodobnostn%C3%AD_algoritmy) (někdy též probabilistické) provádějí některá rozhodnutí náhodně či [pseudonáhodně](https://cs.wikipedia.org/wiki/Pseudon%C3%A1hodn%C3%A1_%C4%8D%C3%ADsla).
* V případě, že máme k dispozici více počítačů, můžeme úlohu mezi ně rozdělit, což nám umožní ji vyřešit rychleji; tomuto cíli se věnují [**paralelní algoritmy**](https://cs.wikipedia.org/w/index.php?title=Paraleln%C3%AD_algoritmy&action=edit&redlink=1)**.**
* [**Genetické algoritmy**](https://cs.wikipedia.org/wiki/Genetick%C3%A9_algoritmy) pracují na základě napodobování biologických [evolučních](https://cs.wikipedia.org/wiki/Evoluce) procesů, postupným „pěstováním“ nejlepších řešení pomocí [mutací](https://cs.wikipedia.org/wiki/Mutace) a [křížení](https://cs.wikipedia.org/wiki/K%C5%99%C3%AD%C5%BEen%C3%AD_(biologie)). V [genetickém programování](https://cs.wikipedia.org/wiki/Genetick%C3%A9_programov%C3%A1n%C3%AD) se tento postup aplikuje přímo na algoritmy (resp. programy), které jsou zde chápány jako možná řešení daného problému.
* [**Heuristický algoritmus**](https://cs.wikipedia.org/wiki/Heuristick%C3%A9_algoritmy) si za cíl neklade nalézt přesné řešení, ale pouze nějaké vhodné přiblížení; používá se v situacích, kdy dostupné zdroje (např. čas) nepostačují na využití exaktních algoritmů (nebo pokud nejsou žádné vhodné exaktní algoritmy vůbec známy).

# Vývojový diagram

Vývojový diagram je druh [diagramu](https://cs.wikipedia.org/wiki/Diagram), který slouží ke grafickému znázornění jednotlivých kroků [algoritmu](https://cs.wikipedia.org/wiki/Algoritmus), [pracovního postupu](https://cs.wikipedia.org/wiki/Workflow) nebo nějakého [procesu](https://cs.wikipedia.org/wiki/Proces). Vývojový diagram obsahuje obrazce různého tvaru (obdélníky, kosočtverce, aj.), navzájem propojené pomocí šipek. Obrazce reprezentují jednotlivé kroky, šipky tok řízení. Vývojové diagramy standardně nezobrazují tok dat, ten je zobrazován pomocí data flow diagramů. Vývojové diagramy jsou často využívány v [informatice](https://cs.wikipedia.org/wiki/Informatika) během [programování](https://cs.wikipedia.org/wiki/Programov%C3%A1n%C3%AD) pro analýzu, návrh, dokumentaci nebo řízení procesu.

# Základní symboly

* šipka — určuje směr zpracování
* svislé nebo vodorovné čáry
* mohou se křížit nebo spojovat
* směr dolů a doprava je prioritní, v tomto případě není nutné použít šipky
* šipky se používají jenom v případě, že tento směr je jiný, nebo když je třeba směr toku informace zvýraznit, například při znázornění iterace
* [obdélník](https://cs.wikipedia.org/wiki/Obd%C3%A9ln%C3%ADk) s popisem — definuje dílčí krok zpracování
* [kosočtverec](https://cs.wikipedia.org/wiki/Koso%C4%8Dtverec) — větvení postupu v závislosti na splnění podmínky, viz [skok](https://cs.wikipedia.org/wiki/Skok_%28informatika%29)
* [obdélník](https://cs.wikipedia.org/wiki/Obd%C3%A9ln%C3%ADk) se zaoblenými rohy — počátek nebo ukončení zpracování
* [kruh](https://cs.wikipedia.org/wiki/Kruh_%28geometrie%29) — spojka několika šipek

# Symboly – graficky

|  |  |
| --- | --- |
| http://programujte.com/galerie/2005/07/200507190835_1.jpg | Konec a začátek algoritmu |
| http://programujte.com/galerie/2005/07/200507190835_2.jpg | Běžný příkaz |
| http://programujte.com/galerie/2005/07/200507190835_3.jpg | Podmíněný výraz |
| http://programujte.com/galerie/2005/07/200507190836_4.jpg | Cyklus s určeným počtem opakování |
| http://programujte.com/galerie/2005/07/200507190836_5.jpg | Cyklus s podmínkou na konci |
| http://programujte.com/galerie/2005/07/200507190836_6.jpg | Cyklus s podmínkou na začátku |
| http://programujte.com/galerie/2005/07/200507190836_8.jpg | Zobrazení výstupu |
| http://programujte.com/galerie/2005/07/200507190837_9.jpg | Zpracování souboru(Tisk) |
| http://programujte.com/galerie/2005/07/200507190842_10.jpg | Uložení dat do souboru |
| http://programujte.com/galerie/2005/07/200507190842_11.jpg | Podprogram |
| http://programujte.com/galerie/2005/07/200507190837_12.jpg | Spojovací značka |
| http://programujte.com/galerie/2005/07/200507190837_13.jpg | Spojovací čára |

# Nakreslete a obhajte vývojový diagram „nalezení nejvyššího čísla z 3“ pro čísla A, B, C.



# Nakreslete a popište u jednoduchého algoritmu funkci podmínky a cyklu se zpětnou vazbou.

**Je už hrnek plný?**

**Nalej vodu do hrnku**

**Načti: Hrnek, Voda**

**Vypít**

# Popište algoritmus obecného přihlášení s kontrolou špatných zadání.

zadání ->

cyklus -> ověření -> rozhodnutí bylo ověření OK ->

=> ANO -> Konec ověřeno

=> NE -> rozhodnutí zda byl dosažen počet chybných zadání

=> NE -> skok na začátek cyklu

=> ANO -> Konec neověřeno