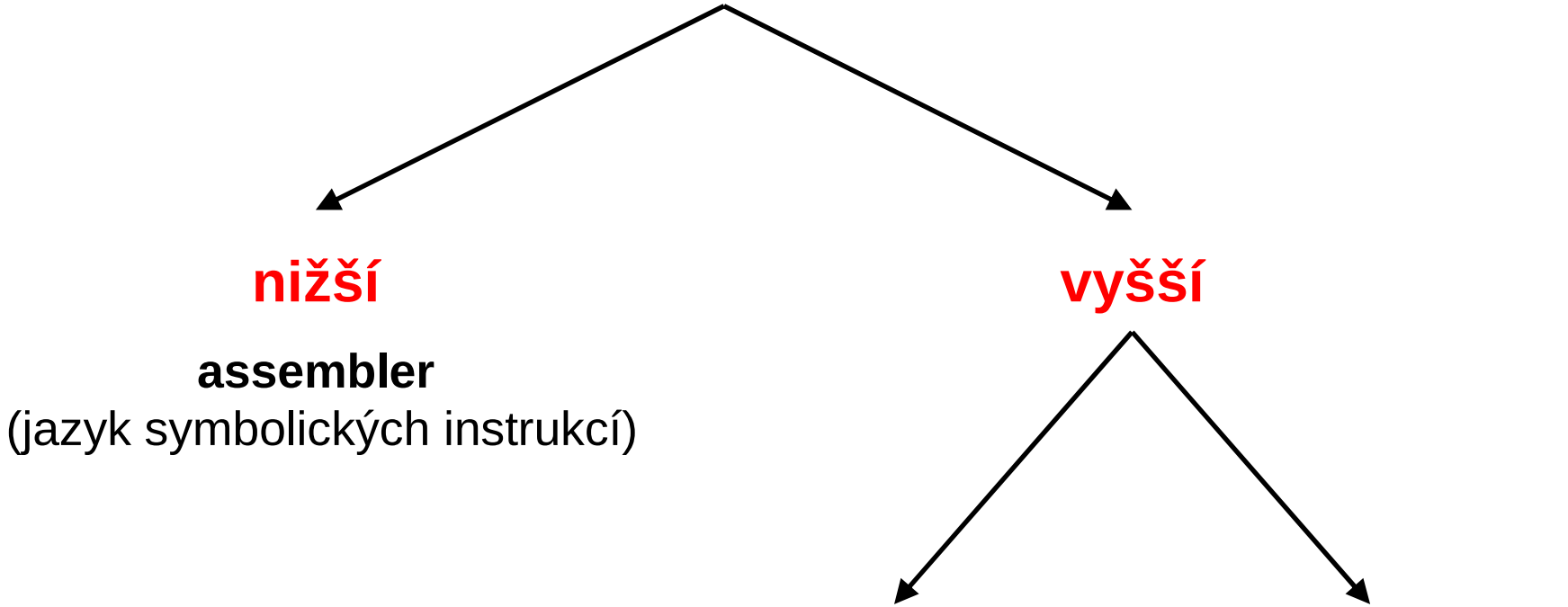
Programovací jazyky

Programovací jazyky



nižší

assembler
(jazyk symbolických instrukcí)

vyšší

imperativní (procedurální)

Pascal, C/C++, Java, Basic, Python, php

neimperativní (neprocedurální)

Lisp, Prolog

Nižší programovací jazyky

- strojově orientované
- příkazy jazyka = instrukce procesoru (zkratky instrukcí) - viz cvičení 1
- výhody:
  - rychlost programu
- nevýhody
  - náročné, složité programování, ne příliš srozumitelný zápis
  - nepřenositelnost na jiné typy procesorů (počítačů)

- použití:
  - dnes spíše výjimečně, např. při psaní rychlých programů pro řídicí jednočipové mikropočítače (i když i tam dnes převládá jazyk C)
  - psaní jader operačních systémů

Příklad:

```
SUB AX,BX
CMP AX,0
JZ pocitej
```

Vyšší programovací jazyky

- formální jazyky (tj. definované formálně, matematicky)
- algoritmus se zapisuje strukturovaně;
  - srozumitelný zápis (anglický, matematický)
- jazyky nejsou závislé na strojových principech počítače (procesoru)
- dělení:
  - imperativní (procedurální, příkazové)
  - neimperativní (neprocedurální).

Imperativní jazyky

- např. Cobol, Pascal, Ada, C/C++, Basic, Java, Javascript, php, Python
- algoritmu se zapisuje posloupností příkazů, cyklů a větvení (podmínek), využívají se proměnné

Příklad:

```
a = a-b;
if (a==0) pocitej();
```

Neimperativní jazyky

Dělení

- **funkcionální**
  - program = množina (ne posloupnost!) funkcí
  - většinou neexistují proměnné, místo nich se pracuje se seznamy dat
  - např. Lisp (AutoLisp v AutoCADu)

- **logické**
  - program = zpravidla množina odvozovacích pravidel, např. logických formulí.
  - využívají se zejména v umělé inteligenci, v expertních systémech.
  - typický představitel - Prolog
    - logický jazyka vestavěný do databází a odvozený od Prologu je Datalog

Ukázka programu v Prologu

- program řeší vztahy mezi členy rodiny

- definujeme klauzule:

```
rodic(Jana,Petr).
rodic(Petr,Eva).
muz(Petr).
zena(Jana).
```

- definujeme odvozovací pravidla:

```
syn(Y,X):-rodic(X,Y),muz(Y).
dcera(Y,X):-rodic(X,Y),zena(Y).
prarodic(X,Z):-rodic(X,Y),rodic(Y,Z).
potomek(Y,X):-syn(X,Y);dcera(X,Y).
```

- můžeme klást dotazy:

```
?-rodic(A,Petr).
?-prarodic(Jana,Eva).
```

- systém odpoví na první dotaz `A=Jana` a na druhý `yes`.
- v Prologu lze psát i výpočty, např. výpočet faktoriálu rekurzí

Kompilační jazyky

- text programu ve vyšším programovacím jazyce = **zdrojový kód**
  - zdrojový kód je vstupem do *překladače*
- překladač (kompilátor, compiler) = program, který převádí zdrojový kód do strojového kódu
  - tj. do formy samostatně spustitelného souboru (soubory .exe nebo .com v operačních systémech firmy Microsoft, ELF formát v systému Linux).

- překladač provádí syntaktickou kontrolu celého zdrojového kódu
  - kontroluje, zda je program zapsán podle pravidel jazyka
  - pokud se vyskytne chyba v zápise programu, je překlad přerušen a seznam chyb je zobrazen uživateli.
- zástupci kompilačních jazyků:
  - Pascal, C/C++

Ukázka:

  - program pro výpočet obsahu kruhu v C

- *výhody:*
  - syntaktická kontrola celého textu najednou, výsledný program možné spustit samostatně bez nutnosti mít speciální programové prostředí
  - běžící samostatný program je rychlý (rychlejší než v případě interpretovaného jazyka.
- *nevýhody:*
  - při jakékoliv změně programu (zdrojového kódu) je nutné provést znovu překlad

Ukázka:

  - program pro výpočet obsahu kruhu v C

Interpretované jazyky

- u interpretovaných jazyků nedochází k samostatnému překladu, nevzniká samostatně spustitelný program
- interpret = program, který *interpretuje* (provádí) zdrojový kód „*řádek po řádku*“
  - interpret přečte jeden řádek zdrojového kódu, provede syntaktickou kontrolu a ihned zajistí jeho provedení (vykonání)

  - na chyby v zápise programu se tedy přijde až při běhu programu (běh programu se v tomto případě přeruší)
- zástupci interpretovaných jazyků:
  - Basic, Python, JavaScript (interpret je přímo zabudován v internetovém prohlížeči), Visual Basic Script, Tcl, neimperativní jazyky

Ukázka:

  - výpočet obsahu kruhu v JavaScriptu

- *výhody:*
  - při změně zdrojového kódu není potřebný překlad
  - pokud existují interprety pro různé operační systémy, pak je snadná distribuce nových verzí programů uživatelům
- *nevýhody:*
  - pomalý běh programů
  - zjištění syntaktických chyb až při běhu

Poznámka:

- dnes i Basic lze kompilovat
- moderní jazyky kombinují oba přístupy