**Programovací jazyky - vlastnosti, srovnání, popis způsobu tvorby i běhu programů**

Jazyk pro tvorbu počítačových programů

* Sada instrukcí pro počítač pro vyřešení problému

Jazyky se liší syntaxí, paradigma, abstrakcí, oblasti použití a způsob překladů

**Co znamená umět programovat?**

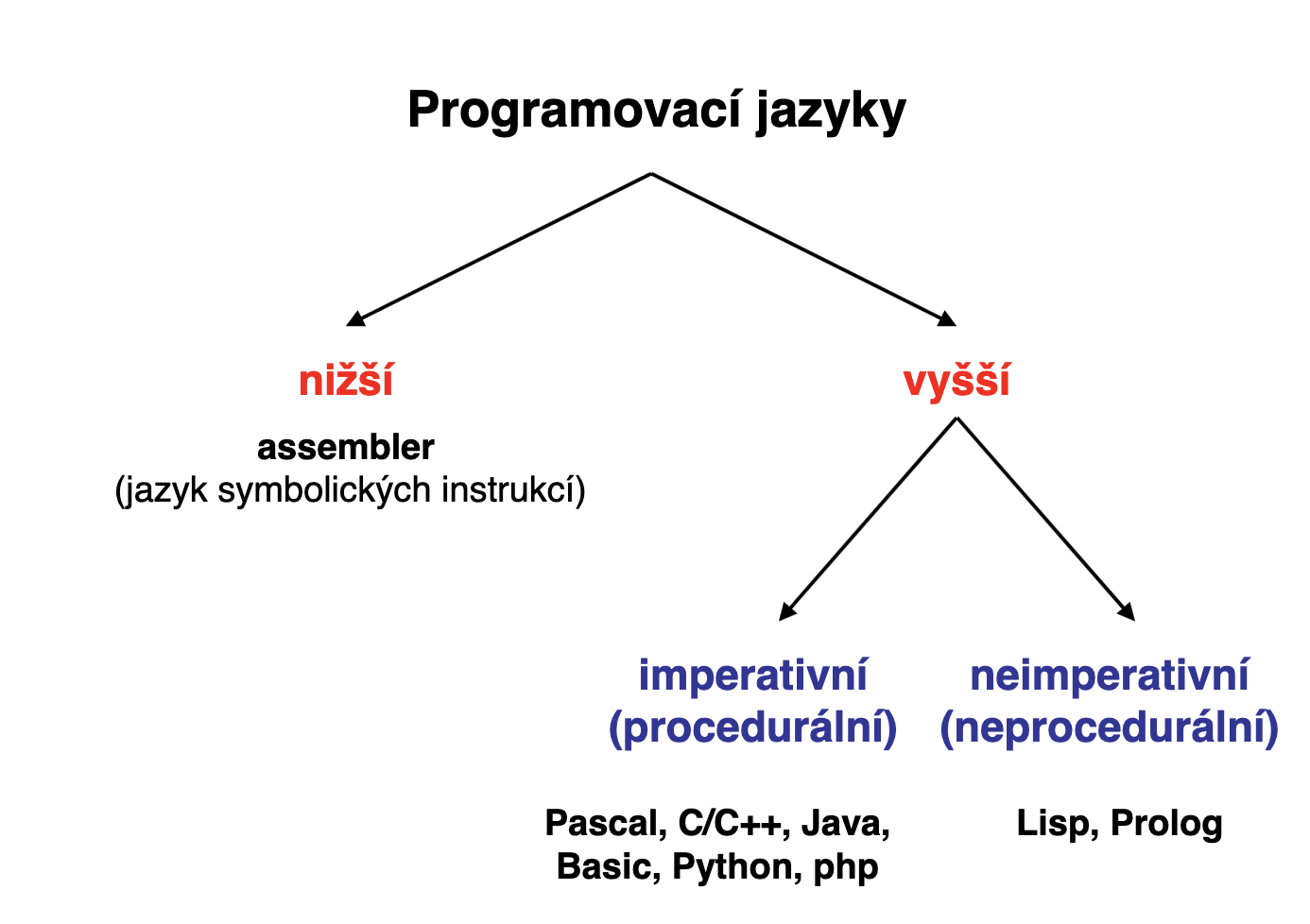
* Schopnost nacházet postupy za účelem řešení problému
* Schopnost formulovat postupy v daném programovacím jazyce
* Znalost syntaxů vůbec nedělá dokonalého programátora

**Jakým jazykem mluví počítač?**

* **Strojový kód** – posloupnost čísel
* Potřeba provést překlad zdrojového kódu programu do strojového kódu procesoru
  + pomocí překladače nebo interpretru
  + Pro každý programovací jazyk je potřeba jiný překladač/interpretr.
* **Překladač**
  + Vytvoří soubor obsahující popis programu ve strojovém kódu
  + Výsledný soubor je spustitelný
  + Pracuje se soubory
* **Interpret**
  + Pro daný příkaz vytvoří překlad do strojového kódu a ihned jej provede
  + Nevytváří výsledný spustitelný soubor.
  + Práce interaktivně
  + Pracuje s jednotlivými příkazys

**Dělení podle abstrakce:**

* **Vyšší programovací jazyky** (většina)
  + Zápis se podobá myšlení člověka (anglicky, matematicky)
  + Tento zápis se kompiluje do strojového kódu (HEX)
    - Počítač nerozumí původnímu zápisu
  + **Procedurální (IMPERATIVNÍ)** (C, C++, Java, Python, PHP)
    - Zápis je posloupnost příkazů, cyklů a podmínek
    - Využití proměnných
    - **Objektově orientované**
      * Základní jednotkou programu je objekt
    - **Strukturované**
      * Např. C
      * Programování pomocí
        + Cyklů
        + Větvení (podmínky)
        + Stuktury PODFUNKCÍ
  + **Neprocedurální** **(NEIMPERATIVNÍ)** (Haskell)
    - Místo tvoření přesného postupu tvoříme jeho cíl, čeho má dosáhnout
    - **Funkcionální**
      * Program sestaven z matematických funkcí
      * Hodně blízke k matematice
    - **Logické**
      * Program sestaven z matematické logice
* **Nižší programovací jazyky**
  + Strojově orientované
  + **Příkazy jazyka = instrukce procesoru**
  + Dělí se na generace
  + Nesrozumitelný/těžko srozumitelný pro člověka zápis (srozumitelný pro počítače)
  + Příklad
    - jazyk symbolických adres (ASSEMBLER)
  + Pracuje v BIN
  + Můžeme manipulovat s HEXADECIMÁLNÍ částí (stejně nesrozumitelná)
  + Assembler – zjednodušení – kompiluje do HEX



Kompilační

* Překládá kód
* Kompiluje hned
* Výsledkem jsou většinou EXE soubory
* Komplexní kód může zabrat hodně času zkompilovat
  + Jsou poměrně rychlý
  + Kompilátor vyhodí chybu – je to hned viděts
* C, C++

Interpretované

* Kompilují pouze zrovna části, které se používají
* Start programu bude rychlejší (nekompiluje vše zároveň)
* Kompiluje za běhu programu
* PŘ. Tlumočník
  + Člověk řekne větu
  + Tlumočník přeloží
* Tzv. částečná kompilace
* PHP
* Výhody?
  + Oproti kompilovanýmu – přenosný mezi platformy
  + Není zkompilovaný pohromadě
  + Datový typ lze měnit během běhu programu

Jazyky s virtuálním strojem

* Máme kompilátor
  + Kompiluje do MEZIKÓDU (je binární)
* Jednodušeji se odhalují chyby
* Datové typy pevně daný
* Kód je přenositelný
* JAVA, C#

**Dělení podle způsobu překladu a kompilace:**

* Kompilační
* Interpretované

**Dělení podle oblasti užití:**

* Univerzální
* Doménově specifické

Neexistuje jazyk, který by byl optimální pro VŠE

Jazyky jsou většinou zaměřeny na oblasti

**Python**

* Práce s daty
* Finance
* Analytika
* Umělá inteligence
* Jednoduchá sytaxe
* Flexibilita
* Facebook, Netflix

**Java**

* SW v bankovnictví
* Průmysl

**Assembler**

* Nejblíže k HW
* Programy přímo řídící HW
* Kernel
* C (mezi nižším a vyšším)

Dělení podle Statických nebo Dynamických typování

Statický

* Dat. Typ určen při deklaraci
* V průběhu programu se dat. Typ nemůže měnit
* Vyhoda
  + Při špatném dat. typu – vyhodí error
* Nevýhoda
  + Komplikace psaní kódu
  + Obcházení změn
* C#, JAVA

Dynamický

* Lze dynamicky měnit datový ty
* Dat. Typ určen hodnotou
* Měněn za běhu programu
* Nelze kontrolovat při kompilaci
  + Až v průběhu se zjistí