



# PROGRAMOVACÍ JAZYKY

Adam Klepáč

7. ledna 2024

# OBSAH

Počítačový program

Programovací jazyky

# Počítačový program

# Co je počítačový program?

Počítačový program je sada instrukcí (uložená ve vnitřní paměti), kterou má CPU vykonat.



# Co je počítačový program?

Počítačový program je sada instrukcí (uložená ve vnitřní paměti), kterou má CPU vykonat. Popis a způsob zápisu instrukcí, které CPU umí vykonat, je určen jeho *instrukční sadou*.

# CO JE POČÍTAČOVÝ PROGRAM?

Počítačový program je sada instrukcí (uložená ve vnitřní paměti), kterou má CPU vykonat. Popis a způsob zápisu instrukcí, které CPU umí vykonat, je určen jeho *instrukční sadou*. Instrukce společné většině instrukčních sad jsou:

- Datové a paměťové
  - **SET** – nastaví registr na konkrétní hodnotu;

# Co je počítačový program?

Počítačový program je sada instrukcí (uložená ve vnitřní paměti), kterou má CPU vykonat. Popis a způsob zápisu instrukcí, které CPU umí vykonat, je určen jeho *instrukční sadou*. Instrukce společné většině instrukčních sad jsou:

- **Datové a paměťové**
  - **SET** – nastaví registr na konkrétní hodnotu;
  - **COPY** (též **MOVE**) – zkopiuje data z registru nebo místa v paměti do registru nebo místa v paměti;

# CO JE POČÍTAČOVÝ PROGRAM?

Počítačový program je sada instrukcí (uložená ve vnitřní paměti), kterou má CPU vykonat. Popis a způsob zápisu instrukcí, které CPU umí vykonat, je určen jeho *instrukční sadou*. Instrukce společné většině instrukčních sad jsou:

- **Datové a paměťové**

- **SET** – nastaví registr na konkrétní hodnotu;
- **COPY** (též **MOVE**) – zkopíruje data z registru nebo místa v paměti do registru nebo místa v paměti;
- **READ/WRITE** – přečti/zapiš data z připojeného/na připojené zařízení.

# CO JE POČÍTAČOVÝ PROGRAM?

Počítačový program je sada instrukcí (uložená ve vnitřní paměti), kterou má CPU vykonat. Popis a způsob zápisu instrukcí, které CPU umí vykonat, je určen jeho *instrukční sadou*. Instrukce společné většině instrukčních sad jsou:

- **Výpočetní a logické**
  - **ADD/SUBTRACT/MULTIPLY/DIVIDE** – provede danou aritmetickou operaci na hodnoty ve dvou registrech a výsledek uloží opět do registru;

# CO JE POČÍTAČOVÝ PROGRAM?

Počítačový program je sada instrukcí (uložená ve vnitřní paměti), kterou má CPU vykonat. Popis a způsob zápisu instrukcí, které CPU umí vykonat, je určen jeho *instrukční sadou*. Instrukce společné většině instrukčních sad jsou:

- **Výpočetní a logické**
  - **ADD/SUBTRACT/MULTIPLY/DIVIDE** – provede danou aritmetickou operaci na hodnoty ve dvou registrech a výsledek uloží opět do registru;
  - **AND/OR/NOT** – provede danou logickou operaci na hodnoty ve dvouregistrech (v případě **AND** a **OR**) nebo jednom registru (v případě **NOT**) a výsledek uloží do registru;

# CO JE POČÍTAČOVÝ PROGRAM?

Počítačový program je sada instrukcí (uložená ve vnitřní paměti), kterou má CPU vykonat. Popis a způsob zápisu instrukcí, které CPU umí vykonat, je určen jeho *instrukční sadou*. Instrukce společné většině instrukčních sad jsou:

- **Výpočetní a logické**
  - **ADD/SUBTRACT/MULTIPLY/DIVIDE** – provede danou aritmetickou operaci na hodnoty ve dvou registrech a výsledek uloží opět do registru;
  - **AND/OR/NOT** – provede danou logickou operaci na hodnoty ve dvouregistrech (v případě **AND** a **OR**) nebo jednom registru (v případě **NOT**) a výsledek uloží do registru;
  - **COMPARE** – porovná hodnoty ve dvouregistrech a výsledek uloží do registru;

# CO JE POČÍTAČOVÝ PROGRAM?

Počítačový program je sada instrukcí (uložená ve vnitřní paměti), kterou má CPU vykonat. Popis a způsob zápisu instrukcí, které CPU umí vykonat, je určen jeho *instrukční sadou*. Instrukce společné většině instrukčních sad jsou:

- **Výpočetní a logické**
  - **ADD/SUBTRACT/MULTIPLY/DIVIDE** – provede danou aritmetickou operaci na hodnoty ve dvou registrech a výsledek uloží opět do registru;
  - **AND/OR/NOT** – provede danou logickou operaci na hodnoty ve dvouregistrech (v případě **AND** a **OR**) nebo jednom registru (v případě **NOT**) a výsledek uloží do registru;
  - **COMPARE** – porovná hodnoty ve dvouregistrech a výsledek uloží do registru;
  - instrukce související s aritmetikou plovoucího bodu (floating-point).

# CO JE POČÍTAČOVÝ PROGRAM?

Počítačový program je sada instrukcí (uložená ve vnitřní paměti), kterou má CPU vykonat. Popis a způsob zápisu instrukcí, které CPU umí vykonat, je určen jeho *instrukční sadou*. Instrukce společné většině instrukčních sad jsou:

- **Řídící**
  - **BRANCH** (též **JUMP**) – přesuň se na jinou adresu v paměti a vykonej instrukce tam;

# CO JE POČÍTAČOVÝ PROGRAM?

Počítačový program je sada instrukcí (uložená ve vnitřní paměti), kterou má CPU vykonat. Popis a způsob zápisu instrukcí, které CPU umí vykonat, je určen jeho *instrukční sadou*. Instrukce společné většině instrukčních sad jsou:

- **Řídící**
  - **BRANCH** (též **JUMP**) – přesuň se na jinou adresu v paměti a vykonej instrukce tam;
  - **CONDITIONALLY BRANCH** – je-li daná podmínka splněna, přesuň se na jinou adresu v paměti a vykonej instrukce tam;

# Co je počítačový program?

Počítačový program je sada instrukcí (uložená ve vnitřní paměti), kterou má CPU vykonat. Popis a způsob zápisu instrukcí, které CPU umí vykonat, je určen jeho *instrukční sadou*. Instrukce společné většině instrukčních sad jsou:

- **Řídící**
  - **BRANCH** (též **JUMP**) – přesuň se na jinou adresu v paměti a vykonej instrukce tam;
  - **CONDITIONALLY BRANCH** – je-li daná podmínka splněna, přesuň se na jinou adresu v paměti a vykonej instrukce tam;
  - **INDIRECTLY BRANCH** – přečti adresu z daného místa (paměť, registr, ...) a vykonej instrukce na této adrese;

# Co je počítačový program?

Počítačový program je sada instrukcí (uložená ve vnitřní paměti), kterou má CPU vykonat. Popis a způsob zápisu instrukcí, které CPU umí vykonat, je určen jeho *instrukční sadou*. Instrukce společné většině instrukčních sad jsou:

- **Řídící**
  - **BRANCH** (též **JUMP**) – přesuň se na jinou adresu v paměti a vykonej instrukce tam;
  - **CONDITIONALLY BRANCH** – je-li daná podmínka splněna, přesuň se na jinou adresu v paměti a vykonej instrukce tam;
  - **INDIRECTLY BRANCH** – přečti adresu z daného místa (paměť, registr, ...) a vykonej instrukce na této adrese;
  - **CALL** – proved instrukce mezi danými adresami v paměti, po vykonání se vrát zpět a pokračuj další instrukcí.

# PROGRAMOVACÍ JAZYKY

# CO JE PROGRAMOVACÍ JAZYK?

Programovací jazyk je způsob zápisu počítačového programu, který je blíže lidskému jazyku než program psaný čistě v CPU instrukcích.

# CO JE PROGRAMOVACÍ JAZYK?

Programovací jazyk je způsob zápisu počítačového programu, který je blíže lidskému jazyku než program psaný čistě v CPU instrukcích. Programovací jazyk (stejně jako lidský) je popsán svoji

- **formou/syntaxí** – způsobem zápisu,

# CO JE PROGRAMOVACÍ JAZYK?

Programovací jazyk je způsob zápisu počítačového programu, který je blíže lidskému jazyku než program psaný čistě v CPU instrukcích. Programovací jazyk (stejně jako lidský) je popsán svoji

- **formou/syntaxí** – způsobem zápisu,
- **sémantikou** – významem slov (posloupností symbolů).

# CO JE PROGRAMOVACÍ JAZYK?

Programovací jazyk je způsob zápisu počítačového programu, který je blíže lidskému jazyku než program psaný čistě v CPU instrukcích. Programovací jazyk (stejně jako lidský) je popsán svoji

- **formou/syntaxí** – způsobem zápisu,
- **sémantikou** – významem slov (posloupností symbolů).

Každý programovací jazyk potřebuje bud' svůj (aspoň jeden) **překladač** (compiler) nebo **interpret** (interpreter), aby v něm psané programy mohly být vykonány CPU jako sady základních instrukcí.

# SYNTAX PROGRAMOVACÍHO JAZYKA

Syntax programovacího jazyka je obvykle definována přes **regulární výrazy** (pro lexikální strukturu) a tzv. **Backusovou-Naurovou formou** (pro gramatickou strukturu).

# SYNTAX PROGRAMOVACÍHO JAZYKA

Syntax programovacího jazyka je obvykle definována přes **regulární výrazy** (pro lexikální strukturu) a tzv. **Backusovou-Naurovou formou** (pro gramatickou strukturu). Příklad syntaxe velmi jednoduchého programovacího jazyka může vypadat takto:

```
expression ::= atom | list\\
atom      ::= number | symbol\\
number    ::= [+ -]?[‘0’-‘9’]+\\
symbol    ::= [‘A’-‘Z’ ‘a’-‘z’].*\\
list      ::= ‘(’ expression* ‘)’
```

# SÉMANTIKA PROGRAMOVACÍHO JAZYKA

Významovou stránku programovacího jazyka je obecně mnohem těžší formalizovat.



# SÉMANTIKA PROGRAMOVACÍHO JAZYKA

Významovou stránku programovacího jazyka je obecně mnohem těžší formalizovat. Obvykle se skládá ze dvou částí:

- **statická sémantika** – omezení na strukturu „vět“, která je obtížné nebo nemožné definovat v rámci syntaxe.

# SÉMANTIKA PROGRAMOVACÍHO JAZYKA

Významovou stránku programovacího jazyka je obecně mnohem těžší formalizovat. Obvykle se skládá ze dvou částí:

- **statická sémantika** – omezení na strukturu „vět“, která je obtížné nebo nemožné definovat v rámci syntaxe.

U většiny programovacích jazyků například udává, že proměnné musejí být inicializovány před použitím nebo že dvě různé větve stejné podmínky musejí být podmíněny různými výroky.

# SÉMANTIKA PROGRAMOVACÍHO JAZYKA

Významovou stránku programovacího jazyka je obecně mnohem těžší formalizovat.  
Obvykle se skládá ze dvou částí:

- **dynamická sémantika** – definuje způsob, jakým se mají interpretovat (v CPU instrukcích) konkrétní věty.

# SÉMANTIKA PROGRAMOVACÍHO JAZYKA

Významovou stránku programovacího jazyka je obecně mnohem těžší formalizovat.  
Obvykle se skládá ze dvou částí:

- **dynamická sémantika** – definuje způsob, jakým se mají interpretovat (v CPU instrukcích) konkrétní věty.

Například udává, že

```
if x < 0:  
    print("x is less than 0.")
```

má podmínečně skočit, kde je uložen program `print("x is less than 0.")`, vykonat ho, pak se vrátit a pokračovat další instrukcí.

# DATOVÉ TYPY

Každé datum, které ukládáte přes programovací jazyk, má konkrétní **typ** – např. celé číslo, desetinné číslo, string, pole apod.

# DATOVÉ TYPY

Každé datum, které ukládáte přes programovací jazyk, má konkrétní **typ** – např. celé číslo, desetinné číslo, string, pole apod.

Datové typy jsou pro moderní programovací jazyky v podstatě nezbytné, protože zjednodušené syntaxe je dosaženo na úkor přesnosti vyjádření instrukcí.

# DATOVÉ TYPY

Každé datum, které ukládáte přes programovací jazyk, má konkrétní **typ** – např. celé číslo, desetinné číslo, string, pole apod.

Datové typy jsou pro moderní programovací jazyky v podstatě nezbytné, protože zjednodušené syntaxe je dosaženo na úkor přesnosti vyjádření instrukcí.

Mnoho programovacích jazyků již docílilo kompletní automatizace správy paměti (např. JavaScript, Python, C# apod.)

# DATOVÉ TYPY

Programovací jazyky můžeme rozdělit do kategorií podle toho, jak zacházejí s datovými typy:

## DATOVÉ TYPY

Programovací jazyky můžeme rozdělit do kategorií podle toho, jak zacházejí s datovými typy:

- **netypované jazyky** – programovací jazyky **bez datových typů**, programátor tedy provádí operace přímo na uložená data (posloupnosti 1 a 0). Sám musí vědět, co daná posloupnost znamená. V podstatě pouze nejstarší jazyky, nebo obecně jazyky velmi blízko strojovému kódu (instrukce psané přímo v 0 a 1), jsou netypované.

## DATOVÉ TYPY

Programovací jazyky můžeme rozdělit do kategorií podle toho, jak zacházejí s datovými typy:

- **netypované jazyky** – programovací jazyky **bez datových typů**, programátor tedy provádí operace přímo na uložená data (posloupnosti 1 a 0). Sám musí vědět, co daná posloupnost znamená. V podstatě pouze nejstarší jazyky, nebo obecně jazyky velmi blízko strojovému kódu (instrukce psané přímo v 0 a 1), jsou netypované.
- **typované jazyky** – programovací jazyky, kde každé v paměti uložené datum má jasně přiřazený typ. Datový typ bývá obvykle uložen v několika prvních bitech samotného data. To znamená, že třeba posloupnost 01001011 bude v programovacím jazyku interpretována podle toho, jaké bity jí předchází.

## TYPOVANÉ JAZYKY

Typované programovací jazyky pak můžeme ještě rozdělit podle toho, jestli nutí programátora typy deklarovat či nikoliv.

## TYPOVANÉ JAZYKY

Typované programovací jazyky pak můžeme ještě rozdělit podle toho, jestli nutí programátora typy deklarovat či nikoliv.

- **staticky typované** jazyky vyžadují u inicializace každé proměnné explicitně vypsat její typ. Jejich syntaxe zakazuje používat operace na datové typy, pro které nebyly definovány.

## TYPOVANÉ JAZYKY

Typované programovací jazyky pak můžeme ještě rozdělit podle toho, jestli nutí programátora typy deklarovat či nikoliv.

- **staticky typované** jazyky vyžadují u inicializace každé proměnné explicitně vyspat její typ. Jejich syntaxe zakazuje používat operace na datové typy, pro které nebyly definovány.
- **dynamicky typované** jazyky nevynucují explicitní vypsání typu a často se typ proměnné za běhu programu mění. Kompatibilita operací s typem proměnné se ověřuje až za překladu nebo interpretace.

# Rozšířené programovací jazyky

1. **Python** (dynamicky typovaný, překládaný i interpretovaný) – víceúčelový programovací jazyk. Používá se hlavně na
  - datovou analýzu a vývoj umělé inteligence,
  - vývoj webových aplikací,
  - testování a simulaci chování softwaru.

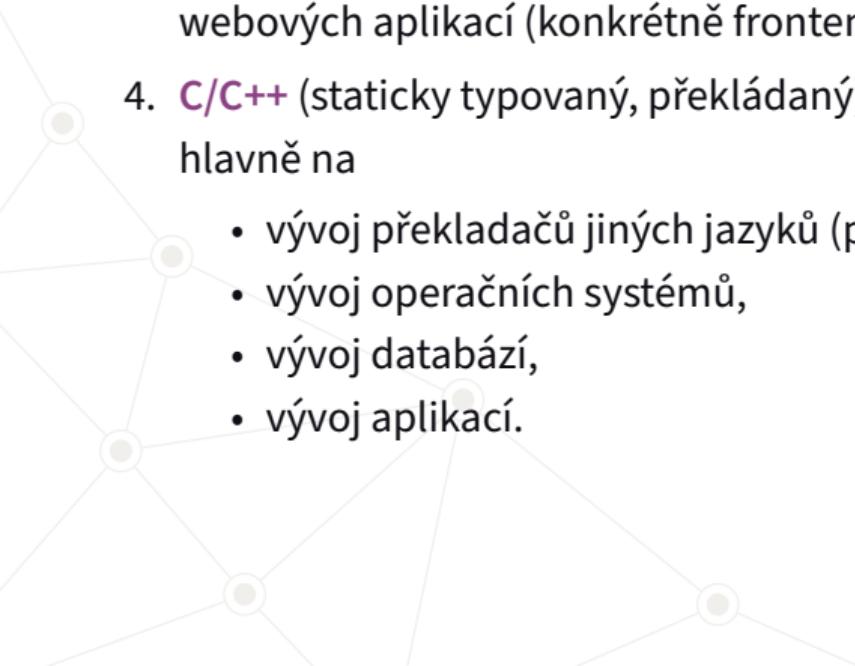
# Rozšířené programovací jazyky

1. **Python** (dynamicky typovaný, překládaný i interpretovaný) – víceúčelový programovací jazyk. Používá se hlavně na
  - datovou analýzu a vývoj umělé inteligence,
  - vývoj webových aplikací,
  - testování a simulaci chování softwaru.
2. **Java** (staticky typovaný, překládaný i interpretovaný) – používaný hlavně na
  - vývoj mobilních aplikací,
  - vývoj clouдовých služeb,
  - chatboty a marketing,
  - rozsáhlé webové aplikace.

# Rozšířené programovací jazyky

3. **JavaScript** (dynamicky typovaný, interpretovaný) – hlavní využití JS je ve vývoji webových aplikací (konkrétně frontendu = toho, co vidí uživatel).

# Rozšířené programovací jazyky

- 
3. **JavaScript** (dynamicky typovaný, interpretovaný) – hlavní využití JS je ve vývoji webových aplikací (konkrétně frontendu = toho, co vidí uživatel).
  4. **C/C++** (staticky typovaný, překládaný) – extrémně rychle překládaný jazyk, používaný hlavně na
    - vývoj překladačů jiných jazyků (překladá například Python),
    - vývoj operačních systémů,
    - vývoj databází,
    - vývoj aplikací.