

Programovací jazyky a jejich syntaxe

Zatímco lidské jazyky jsou tolerantní k drobným chybám (překlepy, chybějící čárky), počítačové jazyky vyžadují absolutní přesnost. **Syntaxe** definuje vnější podobu programu, zatímco **sémantika** definuje jeho význam (co program skutečně dělá).

1. Základní stavební kameny syntaxe

Téměř všechny moderní jazyky sdílejí několik základních syntaktických konceptů:

- **Klíčová slova (Keywords):** Rezervovaná slova, která mají pro jazyk speciální význam (např. `if`, `else`, `while`, `return`, `class`).
- **Identifikátory:** Názvy, které dává programátor proměnným, funkcím a třídám.
- **Operátory:** Symboly pro operace (např. `+`, `-`, `*`, `/`, `==`, `&&`).
- **Literály:** Fixní hodnoty přímo v kódu (např. číslo 42 nebo řetězec „Ahoj“).
- **Oddělovače:** Symboly určující strukturu (závorky `()`, `{}`, `[]` nebo středníky `;`).

2. Různé přístupy ke struktuře kódu

Jazyky se liší v tom, jak určují hranice bloků kódu (např. co patří do cyklu nebo funkce):

Jazyky se složenými závorkami (C-style)

Využívají složené závorky `{ }` k vymezení bloků a středníky `;` k ukončení příkazů.

- **Zástupci:** C, C++, Java, JavaScript, C#, PHP.
- **Příklad (C#):**

```
if (teplota > 30) {  
    Console.WriteLine("Je horko");  
}
```

Jazyky s významným odsazením (Indentation-based)

Struktura kódu je definována mezerami nebo tabulátory na začátku řádku.

- **Zástupci:** Python, Haskell, YAML.
- **Příklad (Python):**

```
if teplota > 30:  
    print("Je horko")
```

*Poznámka: Chybné odsazení v Pythonu vede k chybě (`IndentationError`).

3. Typový systém: Statický vs. Dynamický

Syntaxe se také liší podle toho, jak jazyk zachází s datovými typy:

- **Statické typování:** Programátor musí (nebo kompilátor sám) určit typ proměnné při jejím vzniku. Typ se už nemůže změnit.
 - *Příklad (Java):*

```
int cislo = 5;
```

- **Dynamické typování:** Typ proměnné se určí až za běhu podle toho, co do ní vložíte.
 - *Příklad (Python):*

```
cislo = 5
```

(v dalším kroku klidně

```
cislo = "pět"
```

).

4. Syntaktický cukr (Syntactic Sugar)

Tento termín označuje části syntaxe, které jsou navrženy tak, aby se kód lidem lépe četl a psal, přestože danou věc lze vyjádřit i složitěji.

- **Příklad:** Operátor

```
i++
```

místo

```
i = i + 1
```

.

- **Příklad:** Deklarace vlastností v moderních verzích C# nebo Java, které zkracují dlouhé definice getterů a setterů v [objektovém programování](#).

5. Proces zpracování: Od zdrojového kódu k instrukcím

Aby mohl počítač kód vykonat, musí projít syntaktickou analýzou:

1. **Lexikální analýza (Lexing):** Rozdělení textu na základní jednotky (tokeny).
2. **Syntaktická analýza (Parsing):** Kontrola, zda tokeny tvoří platné struktury podle gramatiky jazyka. Výsledkem je **AST (Abstract Syntax Tree)**.
3. **Generování kódu:** Převod stromu do strojových instrukcí nebo mezikódu.

Související články:

- [Programovací jazyky \(Java, Python, C++\)](#)
- [Objektově orientované programování](#)
- [Algoritmy a datové struktury](#)

Tagy: `dev` `syntax` `programming_languages` `compilers` `parsing` `indentation`

From:

<http://serviceit.cz/> - **IT ENCYKLOPEDIE**

Permanent link:

<http://serviceit.cz/doku.php?id=it:dev:languages>

Last update: **2026/01/02 13:40**

