# Rekurze

**Rekurze** je v informatice mocným nástrojem pro řešení problémů, které mají hierarchickou nebo sebepodobnou strukturu. Aby rekurze neskončila nekonečným cyklem, musí mít vždy jasně definovanou ukončovací podmínku.

## 1. Dvě základní části rekurze

Každá správně napsaná rekurzivní funkce musí obsahovat:

```
1. **Základní případ (Base Case):** Podmínka, při které rekurze končí a vrací konkrétní výsledek. Bez ní by program běžel "do nekonečna" (resp. do pádu aplikace).
2. **Rekurzivní krok:** Část, kde funkce volá samu sebe s mírně pozměněnými (obvykle menšími) parametry, čímž se přibližuje k základnímu případu.
```

## 2. Klasický příklad: Faktoriál

Faktoriál čísla $n$ (značeno $n!$) je součin všech čísel od 1 do $n$. Matematicky lze říci, že $5! = 5 \times 4!$. Tím jsme definovali faktoriál pomocí faktoriálu (rekurze).

**Pseudokód:**

```
funkce faktorial(n):
    pokud n == 0:          // Základní případ
        vrat 1
    jinak:                 // Rekurzivní krok
        vrat n * faktorial(n - 1)
```

## 3. Rekurze v datových strukturách

Rekurze je nejpřirozenějším způsobem, jak procházet složité struktury:

- **Složky v počítači:** Složka obsahuje soubory a další složky. Funkce pro hledání souboru prohledá složku, a pokud najde podsložku, zavolá samu sebe na tuto podsložku.
- **Stromy (Trees):** Algoritmy jako DFS využívají rekurzi k procházení větví grafu.
- **HTML/XML:** Prohlížeče používají rekurzi k vykreslení prvků, které jsou vnořené do jiných prvků.

## 4. Rizika: Stack Overflow

Každé volání funkce spotřebovává místo v paměti zvané **Stack** (zásobník). Pokud je rekurze příliš hluboká (tisíce volání), paměť se zaplní a dojde k chybě **Stack Overflow** (přetečení zásobníku).

| Vlastnost | Iterace (Cyklus) | Rekurze |
|---|---|---|
| **Stav** | Uložen v proměnných. | Uložen v parametrech na zásobníku. |
| **Rychlost** | Rychlejší (menší režie). | Pomalejší (režie volání funkcí). |
| **Čitelnost** | Horší u složitých struktur. | Často mnohem elegantnější a kratší kód. |

# 5. Ocasní rekurze (Tail Recursion)

Některé moderní programovací jazyky (např. Scala, Haskell, ale i některé verze JavaScriptu) umí rekurzi optimalizovat. Pokud je rekurzivní volání **úplně poslední operací** ve funkci, překladač ji převede na běžný cyklus, takže nespotřebovává místo na zásobníku a je stejně rychlá jako iterace.

> **Zajímavost:** Existují tzv. rekurzivní zkratky. Například název projektu **GNU** znamená „GNU's Not Unix“, nebo **PHP** dříve znamenalo „PHP: Hypertext Preprocessor“. Samy sebe definují pomocí sebe sama.

Zpět na Algoritmy