Stack vs. Heap (Zásobník vs. Halda)

Při běhu programu přiděluje operační systém aplikaci paměť RAM, která je logicky rozdělena do dvou hlavních struktur: **Stack** a **Heap**. Správné pochopení jejich fungování pomáhá psát efektivnější kód a předcházet chybám typu „Stack Overflow“.

—

1. Stack (Zásobník)

Stack je paměťová struktura typu **LIFO** (Last-In, First-Out – „poslední dovnitř, první ven“). Funguje podobně jako stoh talířů: nový talíř položíte nahoru a jako první odeberete také ten vrchní.

- **Co se sem ukládá:** Hodnotové typy (integers, bools), lokální proměnné funkcí a adresy pro návrat z funkcí.
- **Správa:** Je automatická a extrémně rychlá. Paměť je alokována při zavolání funkce a uvolněna okamžitě po jejím skončení.
- **Velikost:** Je omezená a relativně malá (obvykle jednotky MB). Pokud dojde k příliš hlubokému vnoření funkcí (např. nekonečná rekurze), dojde k chybě **Stack Overflow**.

—

2. Heap (Halda)

Heap je velký blok paměti určený pro dynamickou alokaci. Data jsou zde ukládána náhodně tam, kde je zrovna volné místo.

- **Co se sem ukládá:** Referenční typy (objekty, pole, instance tříd). Na Stacku zůstává pouze „ukazatel“ (adresa), který míří na data uložená na Heapě.
- **Správa:** Je složitější. V jazycích jako C# nebo Java ji spravuje Garbage Collector, v jazycích C/C++ ji musí spravovat programátor ručně.
- **Velikost:** Je mnohem větší než Stack, omezená prakticky jen velikostí dostupné RAM.

—

3. Srovnávací tabulka

Vlastnost	Stack (Zásobník)	Heap (Halda)
Struktura	LIFO (Stoh)	Hierarchická / Náhodná
Rychlost	Velmi vysoká	Pomalejší (hledání místa)
Správa	Automatická (procesorem)	Garbage Collector nebo ruční
Životnost	Jen po dobu běhu funkce	Dokud je objekt potřeba
Pád aplikace	Stack Overflow	Out of Memory / Memory Leak

—

4. Praktický příklad (C#)

```
public void MojeMetoda() {
    int vek = 25;              // Hodnotový typ -> uložen na STACKU
    string jmeno = "Petr";    // Referenční typ -> "jmeno" (odkaz) je na
STACKU,
                               // ale text "Petr" je uložen na HEAPĚ
} // Po skončení metody "vek" i odkaz "jmeno" okamžitě mizí ze Stacku.
  // Text na Heapě zůstává, dokud ho neuklidí Garbage Collector.
```

—

5. Kdy co použít?

* **Stack** používejte pro krátkodobá, malá data a výpočty uvnitř funkcí. * **Heap** je nezbytný pro data, která musí přežít konec funkce, nebo pro velké objekty, které by se na Stack nevešly.

Související články:

- Garbage Collector
- Přehled datových typů
- Základy OOP a třídy

Tagy: programming memory stack heap performance computer-science