Garbage Collector (GC)

Garbage Collector je komponenta běhového prostředí (runtime), která automaticky spravuje přidělování a uvolňování paměti typu **Heap** (haldy). Díky GC se vývojáři nemusí starat o manuální mazání objektů, což předchází kritickým chybám, jako jsou úniky paměti (memory leaks).

1. Hlavní úkoly GC

- **Alokace paměti:** Vyhledává volné místo na haldě pro nové objekty.
- **Identifikace mrtvých objektů:** Zjišťuje, které objekty již nejsou v programu dosažitelné (žádná proměnná na ně neodkazuje).
- **Uvolňování paměti:** Odstraňuje tyto objekty a vrací paměť operačnímu systému nebo ji připravuje pro další použití.
- **Kompaktizace (Defragmentace):** Přesouvá zbývající objekty k sobě, aby vznikl souvislý blok volného místa.

—

2. Jak GC funguje: Generační hypotéza

Většina moderních GC (zejména v .NET a Javě) pracuje na principu **generací**. Vychází z pozorování, že většina objektů má velmi krátkou životnost (např. dočasné proměnné v metodě).

- **Generace 0:** Nejmladší objekty. GC zde běží velmi často a rychle. Pokud objekt přežije úklid v Gen 0, je povýšen.
- **Generace 1:** Přechodná zóna pro objekty, které přežily první vlnu úklidu.
- **Generace 2:** Objekty s dlouhou životností (např. statická data). Úklid v této generaci je náročný na výkon a děje se méně často (tzv. Full GC).

—

3. Algoritmus Mark-and-Sweep

Toto je základní proces, kterým GC prochází paměť:

```
1. **Mark (Označení):** GC začne od tzv. "kořenů" (roots – např. globální proměnné, zásobník) a projde všechny dosažitelné objekty. Ty označí jako "živé".
2. **Sweep (Smazání):** Všechny objekty, které nebyly označeny, jsou považovány za odpad a jejich paměť je uvolněna.
```

—

4. Výhody a nevýhody

Výhody	Nevýhody
Bezpečnost: Eliminuje chyby typu „Double Free“ nebo „Dangling Pointers“.	**Výkonová režie:** GC spotřebovává procesorový čas pro svou vlastní režii.
Produktivita: Vývojář píše méně kódu pro správu zdrojů.	**Pauzy (Stop-the-world):** Při úklidu může dojít ke krátkému pozastavení běhu aplikace.
Automatická defragmentace: Udržuje haldu v dobrém stavu pro rychlou alokaci.	**Nepředvídatelnost:** Programátor přesně neví, kdy k úklidu dojde.

—

5. Garbage Collector vs. Manuální správa

V jazycích jako **C** nebo **C++** musí programátor paměť uvolňovat ručně pomocí příkazů `free()` nebo `delete`.

- Pokud zapomene: Vznikne **Memory Leak** (aplikace postupně „sežere“ veškerou RAM).
- Pokud smaže příliš brzy: Aplikace spadne při pokusu o přístup k neexistujícím datům.

> **Poznámka:** Ani GC není všemocný. Pokud v programu nechtěně držíte odkaz na velký objekt v globální proměnné, GC ho nesmí smazat, i když ho už nepotřebujete. Tomu se říká „logický únik paměti“.

Související články:

- Přehled datových typů
- Stack vs. Heap (Zásobník vs. Halda)
- Rozhraní IDisposable a blok using

Tagy: programming memory-management garbage-collector dotnet java performance

From:
http://serviceit.cz/ - **IT ENCYKLOPEDIE**

Permanent link:
http://serviceit.cz/doku.php?id=it:sw:garbage_collector

Last update: **2026/01/02 19:01**