**Adresování a správa paměti**

**Garbage collecting, Reference/ukazatele, Struktura paměti programu**

**Adresování/Adresace**

* Určené umístění a přidělení lokace prvku v paměti
* Vyhledání dat počítačem podle dané adresy
* **Fyzická**
  + přenesena na sběrnici
  + fyzicky adresuje hlavní paměť
* **Logická**
  + má ji k dispozici proces
* Druhy přístupu do paměti (paměť programová i datová)
  + **Přímé adresování**
    - V instrukci je přímo uvedeno **ODKUD** nebo **KAM** se data mají přenést
    - Instrukce může být „MOV A, R0, MOV R0“ a je součástí zdrojového kódu
      * V tomto případě se jedná o adresy z paměti RAM
    - Tyto adresy jsou pevně dané
  + **Nepřímé adresování**
    - Můžeme dát pokyn nebo určit adresu
    - Pokyny se mohou dát za běhu programu

**Správa paměti**

* usnadnit fungování operačního systému, ochrany paměti, multitaskingu nebo zvýšit kapacitu paměti
* Soubor metod
* Přidělování operační paměti procesům
* Spolupracuje s Garbage collectingem
* Při spuštění programu se vytvoří tzv. adresní prostor programu
  + Rozdělený na několik částí (V .NET: )
    - Zásobník – lokální hodnotové proměnné
    - Malá řízená halda – malé řízené objekty do 80 KB
      * Tři generace (0,1,2)
    - Velká řízená halda – velké řízené objekty

**Garbage collecting**

* Automatizovaná správa paměti
* Součásti programu nebo prostředí
* Automaticky uvolňuje nepoužívanou paměť
* V programováni = mazání paměti, která byla už nepotřebným objektům
* Zajišťuje aby programu nedošla paměť
* Může mít špatný důraz na výkon protože jeho funkcionalita využívá výpočetní hodnoty
* Objekt je živý do té doby, dokud se používá
* Často objekty třídí do tří kategorii:

1. **Nově vytvořený**. Dočasný objekty se nikdy z této kategorie neposunou, GC je při možnosti umaže
2. **Přeživší první kategorii**. Méně využívaná, objekty, co byli z nějakého důvodu potřebný déle, nebo byli tvořeny s důrazem na jejich životnost se třídí tady.
3. **Dlouho žijící nebo permanentní**. Skoro nepoužívaná kategorie.

* V C# se při „uklízení“ zavolá **Finalizer**
  + Použití pouze u tříd
  + Vola se automaticky
  + Nelze přetížit
  + Nemá parametry

**Ukazatel**

* Odkaz na ADRESU (místo) k proměný
* Datový typ
* Může odkazovat na null
* Uložení adresy v paměti počítače
* Zpřístupnění dat, která jsou uložena na adrese v operační paměti
* Dělí se na typy ukazatelů:
* **Blízký ukazatel**
  + Lineární adresa (offset)
  + Neobsahuje identifikační číslo segmentu
* **Vzdálený ukazatel**
  + Lineární adresa (offset)
  + Obsahuje identifikační číslo segmentu
* V C# je omezený, může mít paměťovou adresu na typy a pole
  + Není zaznamenán Garbage Collectorem
  + V kódu musí být zabalen v „unsafe“

 int \*pt = & x;.

 Console.WriteLine((int)ptr) // Displays the memory address

 Console.WriteLine(\*ptr) // Displays the value at the memory address.

* V Pythonu NENÍ

**Reference**

* Odkaz na PROMĚNOU nebo instanci objektu
* Abstraktní datový typ
* Nemůže odkazovat na null
* Abstraktnější variantou ukazatele, ukazatel nemá žadnou informaci o objektu v operační paměti
* Zvyšují flexibilitu
* Snadnější sdílení mezi kódem
* Je potřeba vhodně používat

**Struktura paměti programu**

* Je sekvenčně seřazená