**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

**Jak zjistit adresu proměnné v paměti v pythonu?**

* Build in metoda ID
* Nelze spoléhat na stálou adresu proměnný – může se v běhu programu měnit
* V pythonu nelze získat fyzickou adresu proměnný (lze získat například v C nebo C++)
* **A picture containing text, font, design, typography

  Description automatically generatedA screenshot of a computer code

  Description automatically generated with low confidence**Adresu taky můžeme přeformátovat do HEXU

**Ale co můžeme s adresou dělat? K čemu nám je?**

* Adresu můžete použít
  + Manuální správa paměti (Místo Garbage collectoru)
  + Zjištění, zda dvě proměnné odkazují na stejnou proměnnou
  + Debuggování
  + Optimalizování – pokud víte, jak funguje adresování a zbytečné deklarování, můžete snadno omezit zbytečné využívání paměti

**Správa paměti**

Je to soubor metod, který dokáže přidělovat a odebírat operační paměť procesům nebo proměnným

**Jak funguje?**

Při spuštění programu se vytvoří tzv. adresní prostor programu

V pythonu je adresní prostor dynamický a nazývá se „virtual memory“

* Tímto lze používat víc paměti, než je vůbec na počítači dostupné
  + Dynamicky přiděluje a odebírá

Cílem správy paměti je

* usnadnit fungování operačního systému, ochrany paměti, multitaskingu nebo zvýšit kapacitu paměti

Spolupracuje s **Garbage collectorem**

**Garbage collecting**

Je to tzv. **automatizovaná správa paměti**

Je většinou 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**
   1. Dočasný objekty se nikdy z této kategorie neposunou, GC je při vyskytnutí šance/možnosti umaže
2. **Přeživší první kategorii**
   1. 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**
   1. Skoro nepoužívaná kategorie

* V pythonu funguje na základě „REFERENCE COUNTING“
  + Počítá počet ukazatelů na objekt
    - Při vzájemném odkazování se počet nedá evidovat
    - Tato metoda dokáže vzájemné odkazy rozeznat, a Garbage collector je může následně vymazat
* V pythonu můžeme odkazy manuálně čistit pomocí metody DEL

A picture containing text, font, screenshot, number

Description automatically generated A computer screen with red and blue text

Description automatically generated with low confidence

**Ukazatel**

Proměnná, která obsahuje odkaz na ADRESU (místo) k jiný proměnný nebo hodnotě

Pomocí ukazatelů mohou programy pracovat s velkými datovými strukturami (pole, řetězce) efektivněji a s menšími nároky na paměť

Dokáže zpřístupnit data, která jsou uložena na adrese v operační paměti

Může odkazovat na NULL

Použití ukazatelů vyžaduje přesnost a pečlivou práci

* Nesmí dojít k přístupu k neplatný adrese
  + To může vést k pádu programu

Je to DATOVÝ TYP

* Velikost ukazatele se liší podle architektury operačního systému
  + 4 bajtů – 32bitový systém
  + 8 bajtů – 64bitový systém

Dělí se na typy ukazatelů:

* **Blízký ukazatel**
  + Neobsahuje identifikační číslo segmentu
* **Vzdálený ukazatel**
  + 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“

**unsafe void Example()**

**{**

**int value = 5;**

**int\* pointer = &value;**

**Console.WriteLine("Value: " + value);**

**Console.WriteLine("Pointer value: " + \*pointer);**

**}**

* V Pythonu NENÍ

**Reference**

Abstraktnější variantou ukazatele, ukazatel nemá žádnou informaci o objektu v operační paměti

* Odkaz na PROMĚNOU nebo instanci objektu
* Abstraktní datový typ
* Nemůže odkazovat na NULL
* Zvyšují flexibilitu
* Snadnější sdílení mezi kódem
* Je potřeba vhodně používat

V python, stačí pouze odkázat na jinou proměnnou

* Změny v první proměnné také ovlivní i tu druhou (ta druhá odkazuje na první)

A screenshot of a computer

Description automatically generated with low confidence A picture containing text, font, typography, screenshot

Description automatically generated

**Struktura paměti programu**

Organizace a správa počítačové paměti

Je to prostor, kde jsou uloženy data a příkazy

Obsahuje hierarchii paměťových kapacit, který se liší jejich prostorem, rychlostí a cenou

Typická hierarchie

(řazená od RYCHLEJŠÍCH + MENŠÍCH do POMALEJŠÍCH + VĚTŠÍCH)

1. REGISTRY
   1. Nejmenší a nejrychlejší
   2. Nachází se v procesoru
   3. Obsahují data
      1. která procesor právě zpracovává
      2. nebo bude za chvíli zpracovávat
2. CACHE
   1. Malá a rychlá
   2. Obsahuje data, ke kterým je často přistupováno
   3. Dělená na několik úrovní (každý jinou rychlost)
3. RAM
   1. Hlavní paměť počítače
   2. Obsahuje data a instrukce na kterých počítač právě pracuje
   3. Promazávají se kompletně při vypnutí
4. VIRTUAL MEMORY
   1. Rozšíření RAM
   2. Používá obyčejnou paměť z hard disku
      1. Pokud je paměť z RAM plná