Implementační dokumentace k 2. úloze do IPP 2023/2024
Jméno a příjmení: Martin Rybnikář
Login: xrybni10

Implementace

Stěžejní metoda mé implementace je ve třídě Interpreter, jedná se o volanou metodu execute(), ve které se přes cyklus do pole nahrají všechny instrukce, které obsahuje XML soubor na vstupu. Každá funkce se stává instancí třídy Instruction, vytvořené přímo pro ukládání instrukce. Každá instance má nula až tři argumenty, podle toho o kterou konkrétní instrukci se jedná. Pro argumenty jsem vytvořil opět speciální třídu. Každý argument ukládá svůj název, typ a hodnotu. Kromě argumentů má instrukce ještě její název, aby se dále vědělo jak se s ní má pracovat.

Po úspěšném načtení instrukcí přijde na řadu druhý cyklus, kde se postupně prochází pole instrukcí, zvolil jsem tuto možnost, protože mi dělala problém instrukce LABEL. Proto jsem vytvořil pomocnou metodu, která najde požadované návěští a vrátí hodnotu indexu, v další iteraci cyklu se začne právě na tomto indexu, ať už je jeho hodnota vyšší nebo nižší než hodnota indexu v předchozí iteraci. V tomto cyklu se volá druhá nejdůležitější metoda celkové implementace, nazval jsem ji loop(). Tato metoda přijímá v parametru aktuální instrukci a porovnává postupně její název s instrukcemi jazyka IPP24. Pokud se v cyklu narazí na neznámou instrukci bude ignorována a přeskočena. Case-insensitive vlastnost jsem vyřešil tak, že všechny názvy instrukcí se převedou na velká písmena. Jakmile dojde ke shodě názvů začne se provádět implementace instrukce. Například instrukce DEFVAR vytvoří neinicializovanou proměnnou na příslušném paměťovém ramci a podobně.

V této hlavní třídě je ještě několik pomocných metod, například na vložení hodnoty na správný paměťový rámec, získání hodnoty ze symbolu (proměnné nebo literálu) nebo získání typu proměnné.

Další významnou třídou je třída Frames, která představuje paměť pro interpretaci, má instance tří pomocných tříd GF_frame, TF_frame a LF_frame. Tyto třídy pak představují jednotlivé paměťové rámce. Jak už se z názvu dá odvodit GF_frame slouží pro globální paměťový rámec. V této třídě jsou metody určené pro práci s paměťovým rámcem. Pro definování proměnné, vložení hodnoty do proměnné, nebo naopak získání její hodnoty nebo typu. Z této třídy zároveň dědí třída TF_frame, neboli dočasný paměťový rámec. Metody zděděné z rodičovské třídy fungují stejně, jen se před jejich provedení kontrolu zda je dočasný paměťový rámec definován. Obsahuje také navíc metody potřebné k vytvoření a zrušení tohoto rámce a převedení lokálního rámce na dočasný. Lokální paměťový rámec je definovaný ve třídě LF_frame, je to samostatná třída a nedědí z žádné jiné třídy. Obsahuje velmi podobné metody jako globální rámec, jen hodnoty ukládá a bere z aktuálně prvního lokálního rámce na zásobníku lokálních rámců, k čemuž také obsahuje nějaké pomocné metody. Je zde i pomocná metoda pro převod dočasného rámce na lokální. Všechny tyto třídy se pak spojí pomocí jejich instancí ve třídě Frames, kde se pak různé pomocné metody pro převod mezi rámcema používají. A zároveň se přes instanci této třídy v hlavní metodě přistupuje k jednotlivým rámcům a jejich metodám.

Méně použitá, avšak také nezbytná třída je třída Text, která konvertuje textový řetězec na požadovaný formát. Jedná se hlavně o escape sekvence. V jazyce IPP24 se používají v decimální soustavě, zatímco jazyk PHP pracuje s hodnotami v oktálové soustavě. Proto tato třída obsahuje metodu na formátování řetězce, kde se jednotlivé hodnoty escape sekvence převedou do oktálové soustavy a až poté na jednotlivé znaky. Poslední mnou implementovaná třída je třída Labels, která obsahuje pole názvů návěští a metodu pro jejich vkládání. Při vkládání se kontroluje, zda už návěští s tímto názvem v poli není vloženo. Pokud nastane kolize program se ukončí s požadovaným chybovým číslem.

Dále jsem použil některé již připravené třídy. To ať už z Core sekce, nebo přímo integrované v PHP. Například třídu SplStack pro práci se zásobníkem, nebo DOMDocument pro práci se vstupním XML souborem.

Diagram tříd