Implementační dokumentace k projektu do IPP 2017/2018

Jméno a příjmení: Jaromír Franěk

Login: xfrane16

1 Parse.php

Jedná se o lexikální a syntaktický analyzátor, který načte ze standartního vstupu zdrojový kód `IPPcode18` a na standartní výstup vypíše jeho XML reprezentaci, pokud byla ověřena lexikální i syntaktická správnost.

Není využit objektově orientovaný přístup, ale strukturovaný. Pro takto jednoduchý program by objektově orientovaný přístup znamenal zbytečně veliký nárůst kódu, ale pouze malý nárůst přehlednosti. Obsahuje pouze dvě třídy, které usnadňovali řešení a umožnili mi vyzkoušet alespoň malé základy objektového programování.

1.1 Parametry

Kontrola vstupních parametrů je prováděna pomocí funkce `checkArg()`, protože jsem nevyužil rozšíření jediným povoleným parametrem je parametr `--help`.

1.2 Analýza vstupu

Analýza standartního vstupu probíhá po řádcích, kde je každý řádek předán funkci `Analyze()`. Ta ověří lexikální a syntaktickou správnost a navrátí instanci třídy token, která udržuje potřebné parametry pro XML generování. Ta je uložena do pole obsahující instance třídy token. Na začátku jsou pomocí regulárního výrazu odstraněny přebytečné bílé znaky a následně komentáře. Zbývající slova jsou rozdělena do pole. První položkou v poli je buď instrukce, nebo bílí znak pokud řádek neobsahoval žádnou instrukci. Každá instrukce je uložena jako klíč v poli a hodnotou je počet operandů. Zkontroluje se, jestli sedí velikost pole s hodnotou v poli, pokud ano pak je provedena kontrola operandů.

1.3 Operandy instrukce

Kontrola operandů je prováděna pomocí regulárních výrazů. Kde se vyhodnotí, zda se jedná o proměnnou, konstantu, typ, nebo label. Hodnoty jsou uloženy do instance třídy token. Následně jsou operandy porovnány s konstantním polem obsahujícím povolené typy operandů. Pokud kontrola prošla je navrácen token beze změny, pokud ne je navrácen token, kdy je typ instrukce nastaven na chybu.

1.4 XML generace

Probíhá pomocí funkce `XMLgenerate()`, kde vstupním parametrem je pole instancí třídy token. Pro vygenerování XML na výstup je využita knihovna `XMLWriter`. Funkce nemá žádnou návratovou hodnotu a po jejím provedení je program ukončen úspěchem.

2 interpret.py

Program slouží k načtení XML reprezentace programu z uvedeného souboru a jeho následné interpretaci, pokud nastane chyba je ukončen s hodnotou chyby.

Je využit objektově orientovaný přístup zpřehledňující kód programu.

2.1 Parametry

Kontrola vstupních parametrů je prováděna pomocí třídy `ArgParser()`, kde jedinými povoleními parametry jsou `--help` a `--source=file`. File je identifikátor souboru, kde se nachází XML reprezentace programu. Žádné další parametry nejsou povoleny.

2.2 Zpracování XML

XML zpracovává třída `XMLreader()`, pomocí knihovny `xml.dom.minidom`. Samotné zpracování je zajištěno metodou `readFile()`, ta zkontroluje kostru souboru a následně volá vnitřní metodu `getProg()`, které prochází program. Procházení probíhá po instrukcích, kde jsou u každé instrukce zpracovány operandy a následně uloženy do instance třídy `Instruction()`. Ta je uložena do pole instrukcí v instanci třídy `Program()`.

2.3 Interpretace programu

Program je interpretován pomocí třídy `Program()`, která si udržuje všechny potřebné atributy, včetně pole instrukcí. Mezi podstatné udržované atributy patří také `PC`, instance třídy pro správu rámců a v nich uložených proměnných, pole s návěštími, zásobník. Pokud je zavolána metoda `do()`, pak se prochází pole instrukcí a každá instrukce se jedna po druhé interpretuje. Samotná metoda `do()` volá metodu `doInst()` a odchytává výjimky.

2.4 Interpretace instrukce

Jednotlivou interpretaci instrukce provádí metoda `doInst()`. Podle `PC` vybere instrukci, která je na řadě. Zkontroluje, zda sedí `opcode` se známými instrukcemi a zavolá metodu pro danou instrukci a předá jí parametry instrukce. Každá metoda dostane potřebný počet parametrů dané instrukce a pokusí se ji vykonat. K vykonání může využít jakýkoliv potřebný atribut. Pokud se vykonání nepovede, pak je vyvolána výjimka, která je odchycena v metodě `do()`. Nakonec inkrementuje hodnotu uloženou na vrcholu zásobníku `PC`.

Pokud metoda pracuje s proměnnými, potom k nim přistupuje pomocí `frames`, to je instance třídy, která udržuje jednotlivé rámce, nebo pole rámců a v nich obsažené proměnné.

Zásobník je implementován formou pole, se kterým se pracuje jako se zásobníkem. Jednotlivé hodnoty jsou na zásobník ukládány jako konstanty, tedy typ a hodnota.

`PC`, tedy programový čítač, je implementován jako pole, se kterým se pracuje jako se zásobníkem. Metody jako `doInst()` využívají hodnotu na vrcholu zásobníku, jednotlivé instrukce např.:`call` mohou na vrchol zásobníku uložit novou pozici v programu podle návěští, na které by se mělo skočit.

3 Test.php

Jedná se o jednoduchý skript pro automatické testování `parse.php` a `interpret.py`. Skript je napsán procedurálně, objektově orientovaný přístup nepřináší v takovémto případě podstatné výhody.

Zpracování parametrů probíhá pomocí funkce `checkArg()`.

Testy jsou vyhledány pomoci funkce `find()`, kde se hledají všechny soubory s příponou: `.src` a pokud zbylé soubory chybí, tak jsou vygenerovány.

Pro porovnání hodnot se využívá nástroj příkazové řádky `diff`.

Následně je vygenerována jednoduchá stránka.