FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Západočeská Univerzita v Plzni

Fakulta Aplikovaných Věd

Formální Jazyky a Překladače

Semestrální práce
GUI pro PL/0 interpret

David Bohmann - A17N0064P
Jakub Váverka - A17N0095P

21. listopadu 2017

Obsah

1 Zadání

Z nabízených možností jsme si vybrali zadání č. 2: Tvorba GUI interpretu pro rozšířený jazyk PL/0.

Cílem práce bylo vytvořit nástroj pro interpretaci rozšířené instrukční sady PL/0, s podporou jeho vlastností. Základní vlastnosti zahrnují zobrazení dat v zásobníku a v registrech (báze, vrchol zásobníku, čítač instrukcí). V rámci rozšířené instrukční sady se jedná o implementaci I/O operací, operace s plovoucí desetinnou čárkou a zobrazení dat na haldě.

Pro implementaci interpretu jsme se rozhodli použít jazyk Java, přičemž tvorba GUI bylo vytvářeno za použití platformy JavaFX.

Repozitář vyvíjené aplikace lze nalézt na GitHubu:
`https://github.com/bombic94/FJP_PL0_GUI`

2 GUI, Struktura aplikace

Grafické rozhraní pro interpret jsme vytvářeli pomocí platformy JavaFX, přičemž jsme kladli důraz na rozdělení aplikace dle architektury *Model-View-Controller*. Striktní rozdělení aplikace dle architektury MVC zaručuje větší bezpečnost, přehlednost kódu a zejména umožňuje jednoduché změny či rozšíření v budoucnu.

2.1 View

JavaFX nabízí pro tvorbu rozhraní nástroj SceneBuilder, který velmi ulehčuje práci a po „naklikání“ vzhledu aplikace vygeneruje výsledný `fxml` soubor, který definuje vzhled aplikace. V tomto souboru je zároveň uvedeno jméno klasické Java třídy, která slouží jako *Controller* a má na starosti logiku aplikace.

Při spuštění aplikace hlavní třída načte `fxml` soubor, zajistí vytvoření *Scene* a zároveň načte `css` soubor, který umožňuje měnit vzhled aplikace. Pro naši aplikaci jsme použili dnes velmi oblíbený *Bootstrap*.

Aplikace je vidět na obrázku 1. V levé části se nachází informace o aktuálním a budoucím stavu zásobníku, napravo leží tabulka instrukcí. Mezi nimi se nachází halda, která se použije pouze v rozšířené instrukční sadě. Ve spodní části se nachází tlačítka po ovládání aplikace a textová pole pro vstup a výstup.

2.2 Controller

Třída `MainController` ovládá komponenty definované v souboru `MainPanel.fxml`. Stará se o vytvoření a naplnění *TableView* a *TreeTableView* daty, definuje chování aplikace na stisk jednotlivých tlačítek a umožňuje vyvolat vyskakovací okno informující o nastalé chybě (čtení souboru, neplatná instrukce, atd.).

Při načtení souboru se o převod z textového souboru do tabulky instrukcí stará třída *FileReader*. V případě, že soubor nedodržuje zadaný formát, je vyvolána výjimka.

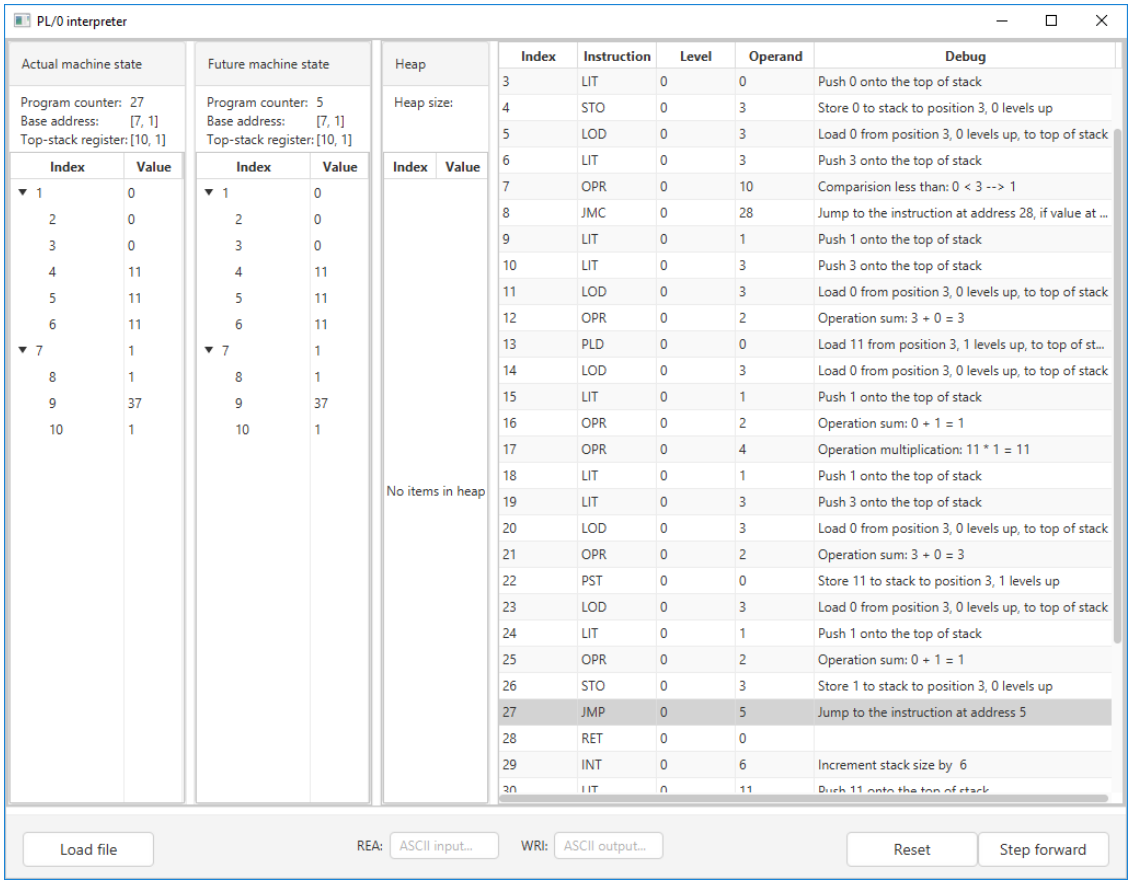
Za běhu programu se o veškeré výpočty stará třída `PL0Debugger`. Zde probíhá zpracování instrukcí, nalezení následující instrukce, vytvoření zásobníku a haldy a provádění jednotlivých operací (viz Tab. 1 a Tab. 2).

`PL0Debugger` obsahuje obrovský switch pro zpracování všech z téměř 20 instrukcí. Kromě toho má velké množství pomocných metod, například pro nalezení nadřazené báze, výpočet indexu v tabulce, a mnoho dalšího (Metody jsou dostatečně okomentované).

2.3 Model

Jednotlivé třídy v balíku *Model* reprezentují položky v tabulce instrukcí, položky v zásobníku a dále je zde třída reprezentující zásobník jako takový. Vzhledem k tomu, že jsme chtěli vytvořit reprezentaci zásobníku pomocí *TreeTableView*, nebyla jeho implementace nejjednodušší.

Následně se zde nachází výčtový typ pro seznam instrukcí (zejména z důvodu bezpečnosti při porovnávání ve switchi), a jednoduchá implementace vlastní Exception, která se stará o vytvoření vyskakovacího okénka a vypsání chyby, která nastala.



Obrázek 1: GUI pro PL/0 interpret

3 Instalační příručka, spuštění

V kořenové složce odevzdaného adresáře se nachází soubor `PL0 Debugger.jar`, který stačí bez parametrů spustit. K běhu programu je potřeba mít nainstalovanou Javu verze alespoň 1.8.

V případě problémů s odevzdaným balíkem lze stáhnout celý repozitář z GitHubu (adresa uvedena výše) a aplikaci posléze vybuildit pomocí Mavenu příkazem `mvn clean package`.

4 Uživatelská dokumentace

Program zvládá zpracovat rozšířenou instrukční sadu PL/0. V přiloženém archivu se nachází mnoho testovacích souborů (složka `resources/testFiles`). Lze tedy použít na interpretaci všech instrukcí, do kterých je program PL/0 přeložen.

4.1 Ovládání

Po spuštění programu se uživateli zobrazí hlavní okno aplikace s prázdnými tabulkami. Kliknutím na tlačítko *Load File* se otevře prohlížeč souborů, který umožní uživateli vybrat textový soubor s tabulkou instrukcí. Pokud je soubor ve správném formátu, proběhne načtení do tabulky instrukcí.

Po načtení tabulky instrukcí lze simulovat běh programu pomocí tlačítka *Step forward*. Při každém kliknutí se program posune o vykonání jedné instrukce dopředu. Kdykoliv v průběhu lze stisknout tlačítko *Reset*, které se postará o „Vynulování“ programu před první instrukci, lze tedy kdykoliv začít interpretovat program od začátku.

Při každém kroku se zvýrazní instrukce, která bude právě vykonávána. Budoucí stav zásobníku reprezentuje stav programu po vykonání zvýrazněné instrukce, zatímco aktuální stav zásobníku reprezentuje stav před vykonáním této instrukce. V zásobníku jsou jednotlivé báze zobrazeny jako kořeny svých podstromů. V rámci vytvořeného aktivačního záznamu jsou na prvních třech položkách - kořen a první dva potomci - uloženy hodnoty potřebné pro operace s hodnotami mimo proceduru a návrat z procedury (odkaz na statickou bázi, dynamickou bázi a čítač instrukcí), poté již následuje prostor pro operace v proceduře.

Zároveň se při zvýraznění instrukce doplní do sloupce *Debug* doplňující informace o tom, co daná instrukce vykonává, přičemž jsou obsaženy informace o aktuálních hodnotách, se kterými operuje. Toto by mělo velmi usnadnit pochopení jednotlivých instrukcí pro běžného uživatele.

Při instrukcích, které operují s haldou, lze pozorovat haldu v tabulce vpravo od tabulek zásobníků. Jedná se jednoduchý seznam.

Pro operace pracující s I/O program pracuje s dvěma textovými poli vespod programu, jedno pro input, druhé pro output. Při čtení program bere první znak textu *Input* zleva, při výpisu přidává znak na konec textu *Output* doprava.

4.2 Seznam instrukcí

V následujících tabulkách jsou uvedeny seznam instrukcí (Tab. 1) a následně pro instrukce `OPR` a `OPF` seznam operací (Tab. 2). U každé instrukce a operace je uveden krátký popis. Pro instrukci `OPF` není implementována operace 6 (modulo) a operace 7 (ověření lichosti), tyto operace mají smysl pouze pro celá čísla.

Tabulka 1: Seznam instrukcí včetně rozšířené instrukční sady

Instrukce	Level	Argument	Popis
OPR	0	A	Provedení operace A
LOD	L	A	Načtení hodnoty z místa určeného L/A
STO	L	A	Uložení hodnoty na místo určené L/A
CAL	L	A	Volání procedury na instrukci A na levelu L
INT	0	A	Zvýšení velikosti zásobníku o A
JMP	0	A	Skok na danou instrukci
JMC	0	A	Podmíněný skok na danou instrukci
RET	0	0	Návrat z podprocedury
REA	0	0	Načtení ASCII hodnoty znaku na inputu
WRI	0	0	Výpis znaku dle ASCII hodnoty na zásobníku
OPF	0	A	Provedení operace A s floaty
RTI	0	0	Převod float na integer
ITR	0	0	Převod integer na float
NEW	0	0	Alokace místa na haldě
DEL	0	0	Dealokace místa na haldě
LDA	0	0	Načtení hodnoty z haldy
STA	0	0	Uložení hodnoty na haldu
PLD	0	0	Dynamické načtení hodnoty z místa určeného L/A
PST	0	0	Dynamické uložení hodnoty na místo určené L/A

Tabulka 2: Seznam operací

Číslo	Operace	Popis
1	Negace	Zneguje číslo na vrcholu zásobníku: a = −a
2	Součet	Součet dvou čísel na vrcholu zásobníku: a = a+b
3	Rozdíl	Rozdíl dvou čísel na vrcholu zásobníku: a = a−b
4	Součin	Součin dvou čísel na vrcholu zásobníku: a = a×b
5	Podíl	Podíl dvou čísel na vrcholu zásobníku: a = a÷b
6	Modulo	Modulo dvou čísel na vrcholu zásobníku: a = a%b
7	Lichost	Ověření lichosti čísla na vrcholu zásobníku a = a%2
8	Rovnost	Porovnání dvou čísel na vrcholu zásobníku: a = b
9	Nerovnost	Porovnání dvou čísel na vrcholu zásobníku: a ≠ b
10	Menší	Porovnání dvou čísel na vrcholu zásobníku: a < b
11	Větší nebo rovno	Porovnání dvou čísel na vrcholu zásobníku: a ≥ b
12	Větší	Porovnání dvou čísel na vrcholu zásobníku: a > b
13	Menší nebo rovno	Porovnání dvou čísel na vrcholu zásobníku: a ≤ b

4.3 Zvláštnosti implementace

Zde bychom se rádi zmínili o několika problémech, kterým jsme čelili v průběhu vývoje, a jak jsme se je rozhodli vyřešit.

Celý program ve skutečnosti funguje o 1 instrukci dopředu, tedy vykonává zrovna to, co se předpokládá jako budoucí stav. Takto byl program navrhnut pro zjednodušení funkce dvou zásobníků, při každém kliknutí o krok dopředu se tedy pouze celý budoucí stav překopíruje do aktuálního stavu. Nicméně po doplnění instrukcí zahrnujících operace s haldou a I/O bylo třeba vyřešit, jak se program bude chovat. Jako nejjednodušší řešení se jevilo, aby halda i vstup a výstup byly také o 1 krok dopředu. Program funguje stále stejně, nicméně je třeba s tímto počítat.

Pokud má proběhnout operace `REA` a na vstupu není žádný znak, bude nahrazen znakem „?“.

Při vykonávání instrukce `CAL` se vytváří nový aktivační záznam, alokace ovšem probíhá až při vykonání instrukce `INT`. Proto se při instrukci `CAL` naplní pomocné pole hodnotami zjištěnými při vykonání instrukce a toto pole se přepíše do zásobníku až při instrukci `INT`. Případně se do zásobníku doplní položky navíc pro běh procedury podle hodnoty argumentu u instrukce `INT`.

Při operacích počítajících s plovoucí desetinnou čárkou je použito zaokrouhlení na dvě desetinná místa. Toto je zaprvé z důvodu zvláštního chování desetinných čísel, kdy například: $4.0 - 3.1 = 0.8999999999999999$, zadruhé z důvodu nešikovného dělení, například $\frac{10}{3}$. Zaokrouhlením na dvě desetinná místa je zaručeno, že do zásobníku se uloží hodnota maximálně 99.

5 Závěr

Práce splňuje body uvedené v zadání. Kromě základní funkcionality nabízí implementaci rozšířené instrukční sady (I/O operace, operace s floaty a operace s haldou). Pro lepší pochopení jednotlivých instrukcí jsou při běhu generovány doplňující informace, které obsahují aktuální hodnoty.

Při tvorbě jsme brali v úvahu to, že práce bude s největší pravděpodobností použita v dalších letech při výuce, proto jsme se snažili ji udělat co nejintuitivnější. Ovládání pomocí co nejmenšího množství tlačítek a zablokování klikání do tabulky instrukcí by mělo pomoci zabránit malfunkcím programu.