Formální jazyky a překladače

KIV/FJP

Ceska Java

Autor
Jan Rychlík, A19N0074P
rychlikj@students.zcu.cz

5. ledna 2021

Obsah

1 Zadání

Cílem semestrální práce z předmětu FJP je seznámit se s jazykem PL0, poté navrhnout vlastní programovací jazyk, sestavit k němu gramatiku a vytvořit překladač právě do instrukcí jazyka PL0. Cílové instrukce nemusí být v jazyce PL0, je možné zvolit libovolnou instrukční sadu, pro kterou existuje interpret.

Jazyk musí obsahovat tyto náležitosti:

- definice celočíselných proměnných
- definice celočíselných konstant
- přiřazení
- základní aritmetiku
- alespoň jeden cyklus
- podmínku if
- definice podprogramu(metoda)

Překladač, který obsahuje tyto základní věci je hodnocen 10 body. Další body lze získat rozšířením o prvky vypsaných v seznamu v nezkráceném zadání. V tomto zpracování jsou vybrány následující:

- zbylé cykly bez for-each 3b
- else, else if větev 1b
- datový typ boolean 1b
- rozvětvená podmínka 1b
- násobné přiřazení 1b
- parametry předávané hodnotou 2b
- návratová hodnota podprogramu 2b

S těmito rozšířeními by měla být práce ohodnocena minimálně 21 body. Další body je možné získat za dokumentaci, správu gitu a strukturu kódu. Kompletní, nezkrácená verze zadání je zde:

https://phix.zcu.cz/moodle/mod/page/view.php?id=186301.

2 Implementace

Projekt obsahuje dva hlavní balíky. Main a compilator. V Balíku main se nachází gramatika jazyka a všechny soubory vygenerovane pomocí antlr4. Druhý balík je roztříděn do následujících bloků:

- compilators - balík obsahující kompilátory pro generování instrukcí
- expressions - balík, který obsahuje třídy pro jednotlivé výrazy
- instructions - balík, který obsahuje všechny třídy pro práci s instrukcemi
- methods - balík, který obsahuje všechny třídy pro práci s metodami
- statements - balík obsahující všechny statementy, které mohou vzniknout v programu
- variable - balík pro práci s proměnnými
- visitors - balik obsahuje všechny visitory pro průchod derivačním stromem.

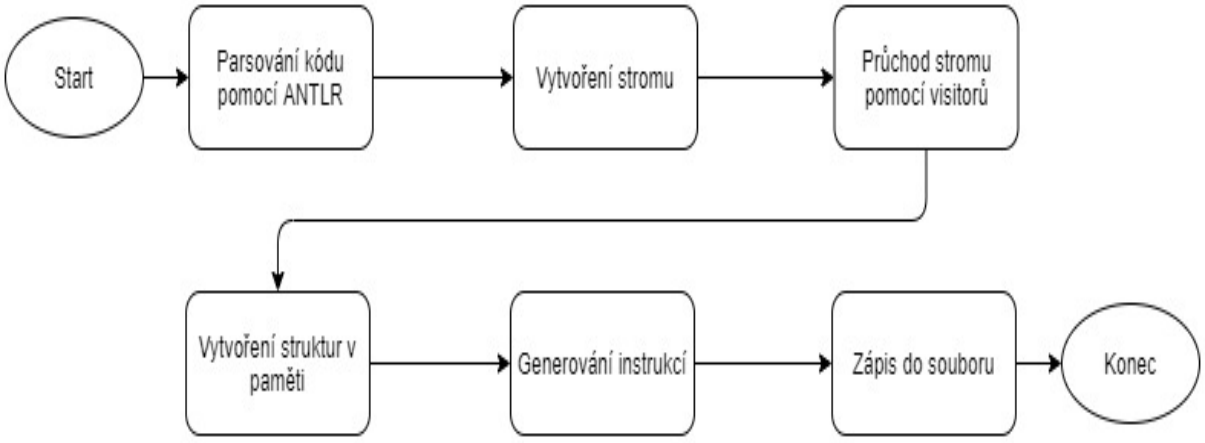
2.1 Funkčnost

Metoda main.java volá třídu compile, ve které je řízen překlad zdrojového souboru. Překlad programu do instrukcí PL0 je definován dvěma průchody. Poprvé, kde se derivační strom ukládá do struktur v paměti a druhým průchodem se vygenerují instrukce. Průběh zpracování je vidět na obrázku 1 . Díky dvojitému průchodu stromem je možné definovat metody kdekoliv v programu. Při generování instrukcí jsou uloženy na konci a odkazuje na ně adresa z programu.

3 Popis tříd

3.1 Průchod stromu pomocí visitorů

Jednotlivé visitory jsou uloženy v baliku compilator.visitors. Jejich rozdělení je definováno tak, aby odpovídaly hlavním uzlům při průchodu derivačním stromem.



Obrázek 1: Zpracování derivačního stromu

3.2 Struktury v paměti

Slouží pro uložení derivačního stromu v paměti po prvním průchodu. Názvy jednotlivých struktur odpovídají uzlům v gramatice. Hlavní objekt Program je výsledkem průchodu a představuje kořen derivačního stromu. Také obsahuje odkazy na instance nižších objektů (uzlů), které odkazují na další.

3.3 Generování instrukcí

Hlavní třídou pro generování instrukcí je třída compilator, ve které jsou uloženy hlavní proměnné a metody. Dědí od něj všechny ostatní třídy z balíku compilátor. Další třídou je blockStatmentCompilator, která na základě typu statementu rozhodne, kterou třídu využije pro generování jednotlivých instrukcí. Každá instrukce je reprezentována instancí třídy Instruction a je uložena do pole pro uložení do souboru.

3.4 Metody

Instrukce pro generování metod se generují nejprve z prototypu metody(hlavičky), z důvodu určení návratové hodnoty při generování instrukcí. Protože metody mohou být definovány kdekoliv v kódu, je u nich nutné nastavit adresu až zpětně. V cyklu se projdou všechny instrukce, pokud se narazí na instrukci CALL, otestuje se, zda je v tabulce symbolů záznam o metodě a následně podle tabulky nastavena adresa. Pokud je metoda s návratovou hodnotou, je obsah zásobníku instrukcí navýšen o 1. Tato adresa odpovídá místu, na kterou je pak uložena návratová hodnota funkce.

3.5 Uložení instrukcí

Všechny instrukce se nejprve ukládají do pole a po vygenerování všech instrukcí se následně uloží do souboru, který je definován jako třetí parametr při spuštění programu.

4 Uživatelská dokumentace

4.1 Překlad

Pro překlad programu je nutné mít nainstalovaný maven, který se postará o jednotlivé zavislosti programu a následné sestavení do spustitelného jar souboru. Provede se příkazem :

Listing 1: Překlad programu

```
1 mvn clean install
```

Poté maven vytvoří spustitelný .jar soubor ve složce target. Pro spuštění je nutné pracovat s programem "ceskaJava-jar-with-dependencies". Překlad vlastního zdrojového soubor se vykoná pomocí příkazu:

Listing 2: Spuštění programu

```
1 java -jar ceskaJava-jar-with-dependencies.jar
2 n zev\_vstupn ho\_souboru n zev\_v stupn ho\_souboru
```

Po skončení běhu programu je v názvu výstupního souboru uložen přeložený program v PL0 instrukcích.

4.2 Konstrukce jazyka

4.2.1 Deklarace proměnných

Veškeré proměnné se dají použít v bloku, ve kterém byly definovány nebo v bloku

Listing 3: Deklarace proměnných

```
1 {
2 konstanta pravdivost a = b = pravda;
3 konstanta pravdivost c = d = lez;
4 cislo g = 19;
5 konstanta cislo j = -7;
6 pravdivost p = pravda;
7 cislo q = g-5;
8 }
```

4.2.2 Větvení

Listing 4: Větvení

```
{
pokud(p && pravda){
a = 30;
}
jinak pokud((a == 5) && pravda){
a = 20;
}
jinak pokud(a == 6){
a = 20;
}
jinak{
a = 100;
}

// switch
        nazaklade (a){
                jestlize 5:{
                        c = a+b;
                }
                jestlize 5:{
                        c = a+b;
                }
                jestlize 10:{
                        c = a-b;
                }
                jestlize 15:{
                        c = a+b+b;
                }
                default :{
                        c = 1;
                }
}
```

4.2.3 Cykly

Listing 5: Cykly

```
{
    //while
        toc(i < 5){
        i = i+1;
        }
        //DoWhile
        pro{
        i = i-1
        }toc(i>0)
        //for
        tocpro(j = 2...8){
                i = j;
        }
        //Repeat-until
        opakuj{

        i = i +2;
        }dokud(i >= 10)
}
```

4.2.4 Metody

Listing 6: Metody

```
{
    test();

        funkce test(){
                cislo beta = 2048;
        }

        pravdivost p1 = vratTrue(lez)
        pravdivost funkce vratTrue(pravdivost l){
                pravdivost p = pravda;
                vrat p;
        }
}
```

5 Závěr

Zadání semestrální práce bylo v daném rozsahu splněno. Je vytvořen překladač vlastního jazyka ceskaJava, ve kterém lze psát základní programy. Dále byly vytvořeny testovací soubory, ve kterých jsou příklady jednotlivých vlastností programu, na kterých byl i celý projekt otestován. Automatické testy vytvořeny nejsou. Dále by bylo vhodnější upravit vypisování chybových hlášek.

Programovací jazyk ceska java vychází z klasické javy. Protože však nabízí mnohem menší funkčnost než jeho vzor pravděpodobně se jazyk nikdy neujme. Další negativní vlastností jsou klíčová slova v češtině. Tento mínus se minul s původním záměrem autora. Bylo očekáváno mnohem vtipnější zpracování, které by na místo těžkého praní kódu proměnilo v téměř nemožné. Například záměna slova 'pravda' a 'true'. Odhadovaná doba strávená nad semestrální prací je přibližně 100-120 hodin a celá práce je na adrese https://github.com/davidbarta/ceskaJava