Semestrální práce
z předmětu KIV/FJP
Překladač do jazyku PL/0

Duc Vuong Tran - tranv@students.zcu.cz
Michal Sakáč - seky@civ.zcu.cz

7. ledna 2018

Kapitola 1

Úvod

Zadání

Zadáním práce bylo vytvořit překladač pro náš definovaný jazyk, který bude tento jazyk překládat do *PL/0* instrukcí. Jazyk musí splňovat určité minimální instrukce a následně je možné implementovat rozšíření jazyka.
Minimální požadavky jazyka:

- Definice celočíselných proměnných
- Definice celočíselných konstant
- Přiřazení
- Základní aritmetiku a logiku
- Libovolný cyklus
- Jednoduchá podmínka
- Definice funkce a jeho volání

1.1 Programovací jazyk PL/0

Jazyk *PL/0* byl navržen profesorem *Niklause Wirthem* jako model, který sloužil pro výklad základních principů překladačů.
V jazyce *PL/0* je definovaný pouze celočíselný datový typ, který se při deklaracích proměnných neuvádí. Program se skládá z bloků, které mohou být do sebe zanořené, také známé jako podprogramy.

1.1.1 Instrukční sada PL/0

- *lit 0, A* - ukládání konstantu A do zásobníku
- *opr 0, A* - provádění instrukci A
  - 1 - unární minus
  - 2 - sčítání
  - 3 - odčítání
  - 4 - násobení
  - 5 - celočíselné dělení
  - 6 - dělení modulo
  - 7 - test, zda je číslo liché
  - 8 - test rovnosti
  - 9 - test nerovnosti
  - 10 - menší než
  - 11 - větší nebo rovno
  - 12 - větší než
  - 13 - menší nebo rovno
- *lod L, A* - načte hodnotu na adrese L,A a uloží jí na vrchol zásobníku
- *sto L, A* - uloží hodnota na vrcholu zásobníku na adresu L,A
- *cal L, A* - zavolá proceduru A
- *int 0, A* - zvýší obsah zásobníku o hodnotu A
- *jmp 0, A* - provádí nepodmíněný skok na adresu A
- *jmc 0, A* - provádí podmíněný skok na adresu A, je-li hodnota vrcholu zásobníku je 0
- *ret 0, 0* - návrat z procedury

1.2 ANTLR

ANTLR je zkratkou pro *ANother Tool for Language Recognition*. Je to mocný nástroj nejen pro práci s gramatikou, ale také dokáže generovat parser pro vytváření a procházení parsovacího stromu.
ANTLR umožňuje generovat libovolnou část z prvních třech částí překladače. Dokáže tedy vytvořit lexikální, syntaktický nebo sémantický analyzátor. Poskytuje, na rozdíl od ostatních generátorů, jediný konzistentní zápis, kterým je možné definovat danou část překladače. Díky této vlastnosti je jeho používání jednodušší.

Kapitola 2

Struktura jazyka a programu

Jazyk byl inspirován jazykem *Pascal* a *PL/0*, který dále obsahuje některé vlastnosti od jiných jazyků, jako například *Java*, *C*.

2.1 Struktura jazyka

Struktura jazyka je definována gramatikou, která se nachází v souboru *Exp.g4*.

2.1.1 Základní podmínky jazyka

- Definice celočíselných proměnných
- Definice celočíselných konstant
- Přiřazení
- Základní aritmetiku a logiku
- Libovolný cyklus (for cyklus)
- Jednoduchá podmínka
- Definice funkce a jeho volání

2.1.2 Rozšíření

- Datový typ *boolean* a logické operace s ním - Datový typ *boolean* je implicitně přetypován do celočiselného datového typu. Tedy lze na něj používat jak základní aritmetické operátory (sčítání, odčítání, násobení a dělení), tak i booleanovské (*AND, OR*). Dokonce lze aplikovat také operátory pro porovnání (větší, menší, apod.)
- Else větev pro podmínku *if*
- Cyklus *While*
- Cyklus *Do-While*
- Cyklus *Repeat-Until*
- Rozvětvená podmínka - *Switch-case*
- Násobné přiřazení - (*a := b := 5;*)
- Pomíněné přiřazení/ternární operátor
- Paralelní přiřazení - ({*a, b*}={*5, 6*};)
- Parametry předávané hodnotou - S maximálním počtem parametrů 3
- Návratová hodnota podprogramu
- Unární operátor - (++a;)

2.2 Struktura programu

Jednoduchý program napsaný v tomto jazyce začíná typickou definicí třídní (globální) proměnné, následně jednotlivé podprogramy a končí znakem tečka ('.').
Hlavička podprogramu se skládá z klíčového slova *function*, za kterým následuje název daného podprogramu, jednotlivé parametry, typ návratové hodnoty, definice lokálních proměnných a potom jednotlivé příkazy. Definice lokálních proměnných na začátku nabízí programu větší přehlednost.
Podprogram musí obsahovat na konci příkaz *return*, který může být prázdný, pokud typ návratového podprogramu je *void*. Důležité je potom mít podprogram *main* jako poslední podprogram.
Tato struktura byla navržena na začátku vývoju a byla postupně upravována do finálního tvaru tak, aby fáze implementace byla co nejjednodušší a aby s v projektu orientovali všichni členy v týmu.

2.3 Struktura projektu

Projekt je rozdělen do několik balíčku, přičemž nejvýznamnější balíčky jsou:

- *types* - Obsahuje jednotlivé objekty reprezentujicí jednotlivých prvků jazyku
- *visitors* - Obsahuje implementace způsob procházení jazyku a následně vytvoří strom objektů, ze které se bude aplikovat proces generování kódů.
- *generator* - Pro generování ze získaného stromu seznam *PL/0* instrukcí.

Po generováním parseru pomocí knihovy *ANTLR* proběhne procházení parser stromu pomocí *visitor* třídy (*visitory*). V těchto třídách se nachází metody pro procházení jednotlivých částí stromu a následně si uloží získané informace do objektů ve zmíněném balíčku *types*.
Po dokončení získáme strom objektů s jedním hlavním kořenem třídy *types/-Program*. Na tohoto objektu se poté aplikuje generátor (třída *generator/CodeGenerator*), který vygeneruje výsledný seznam *PL/0* instrukcí.

2.4 Ukázka

```
int a := 1;
int b := 2;
bool c := true;
function fce(int x): int:
{
    ++x;
    return x;
}
function main(): void:
{
    while(b > a){
        ++a;
    }
    b := 1 + call fce(a) - call fce(b);
    c := b;
    return;
}
.
```

Listing 2.1: Ukázka programu

```
int a := 1;
function main(): void:
 int b := 10;
 int c := 0;
{
    while(b > a){
        ++a;
    }
    a := 0;
    do{
        ++a;
    } while(a > b);
    a := 0;
    repeat{
        ++a;
    } until(a > b);
    for(a < b; ++a){
        ++c;
    }
    return;
}
.
```

Listing 2.2: Ukázky cyklu

Kapitola 3

Použité technologie

Vývoj projektu proběhl na operačním systému *Ubuntu 17.10 x64* s použítím těchto technologii:

- Java 8 - Kompletní práce byla napsána v jazyce *Java*
- IntelliJ IDEA Community Edition - Vývojové prostředí
- Antlr - Nástroj pro lexikální a syntaktickou analýzu
- Github - Správa projektu a jeho verzí
- Texmaker - Tvorba dokumentace

Kapitola 4

Uživatelská dokumentace

4.1 Odkaz na Git

Projekt je uložen jako repozitář na serveru *github.com*. Odkaz na repozitář:

```
https://github.com/seky739/FJP
```

4.2 Příprava

Ke spuštění projektu je třeba mít tyto soubory v jednom adresáři:

- *FJP.jar* - Zabalený spustitelný soubor
- *run.bat* - Slouží ke spuštění programu s defaultním nastavením
- *test.txt* - Vstupní soubor ke spuštění programu pomocí skriptu *run.bat*

Jelikož je součástí balíku spustitelný soubor *FJP.jar*, lze program v přikazovém řádku ručně spustit pomocí příkazu:

```
java -jar FJP.jar vstup.txt
```

4.3 Vstupní soubor

Aplikace přijme do vstupu textový soubor obsahující zdrojového kódu jazyka, který je definovaným již zmíněnou gramatikou.
Soubor pojmenujme jako *test.txt* a zkopírujeme ho do kořenového adresáře.

4.4 Spuštění

Program lze spustit kliknutím na soubor *run.bat*

4.5 Výstup

Po úspěšné spouštění se nám v adresáři vytvoří výstupní soubor s jménem *out_test.txt*, který obsahuje seznam výsledných *PL/0* instrukcí.

Kapitola 5

Závěr

Práce byla dokončená mnohem později než se očekavalo, kvůli neaktivitě člena v týmu (*Michal Sakáč*) z osobních důvodů. Existují různé funkcionality, které se nestihly naimplementovat a musely být vynechány, konkrétněji - pole primitivních datových typů, datový typ pro reálné číslo *ratio* a explicitní přetypování. Během práce jsme využili znalosti získané z předmětu *KIV/FJP*. Naučili jsme se pracovat s nástrojem *ANTLR*, jako vytvořit vlastní překladač a také práci v týmu.