FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Semestrální práce z předmětu KIV/FJP

Překladač jazyka do PL/0

Ondřej Drtina
A20N0077P
drtinao@students.zcu.cz

Eliška Mourycová
A20N0061P
emouryc@students.zcu.cz

9.1.2021

Obsah

1 Zadání **2**
1.1 Zvolená rozšíření 2

2 Uživatelská příručka **3**
2.1 Překlad a spuštění 3

3 Řešení **4**
3.1 Návrh jazyka 4
3.1.1 Struktura jazyka 4
3.1.2 Ukázky použití 4
3.1.3 Omezení jazyka 4
3.2 Struktura projektu 5
3.3 Generování struktur 5
3.4 Tabulky symbolů 6
3.5 Volání procedur 6
3.6 Cyklus foreach 6

4 Závěr **7**

1 Zadání

Zadáním práce byla tvorba vlastního překladače. Zvolili jsme a navrhli gramatiku pro jazyk připomínající C, který je překládán do instrukcí PL/0.

Jazyk musí mít následující konstrukce:

- definice celočíselných proměnných
- definice celočíselných konstant
- přiřazení
- základní aritmetiku a logiku (+, -, *, /, AND, OR, negace a závorky, operátory pro porovnání čísel)
- cyklus (libovolný) - (while)
- jednoduchou podmínku (if bez else)
- definice podprogramu (procedura, funkce, metoda) a jeho volání

1.1 Zvolená rozšíření

Jako rozšíření jsme zvolili následující konstrukce:

- další cykly - do while, for, foreach, repeat until
- datový typ boolean a logické operace s ním
- datový typ string (s operátory pro spojování řetězců)
- násobné přiřazení (a = b = c = d = 3;)
- podmíněné přiřazení / ternární operátor (min = (a < b) ? a : b;)
- pole a práce s jeho prvky

2 Uživatelská příručka

Program při spuštění očekává jeden argument - cestu k souboru se zdrojovým kódem jazyka.

2.1 Překlad a spuštění

Pro překlad je potřeba mít nainstalovaný MAVEN. Automatický překlad a spuštění lze provést pomocí souboru `sampleTrans.bat` - budou vygenerovány instrukce pro jeden ze souborů v adresáři testFiles.

Pro vlastní specifikaci souboru zadejte v kořenovém adresáři příkazy:

```
> mvn clean install
> java -jar target/zcu.fav-1.0-SNAPSHOT-jar-with-dependencies.jar
cesta_k_souboru
```

Výsledné instrukce budou uloženy v souboru `resultInstr.txt`.

3 Řešení

Tato kapitola popisuje navržený jazyk, kostrukce, které podporuje, jeho omezení a ukázky použití. Dále se stručně věnuje technickým aspektům implementace.

3.1 Návrh jazyka

3.1.1 Struktura jazyka

Jazyk svoji gramatikou připomíná zjednodušený jazyk C. Popis gramatiky sa nachází v souboru `ourC.g4`, který se nachází v kořenovém adresáři odevzdávaného archivu.

Soubor se zdrojovým kódem musí obsahovat proceduru s názvem `main`. Veškerý výkonný kód (resp. odpovídající instrukce) bude generován z této procedury (kromě globálních deklarací).

3.1.2 Ukázky použití

Ukázky kódu jsou k nalezení v adresáři `testFiles`.

3.1.3 Omezení jazyka

- Proměnná typu string musí být deklarována a zároveň inicializována pomocí literálu
- String nebo pole není možné prodloužit
- Do proměnné typu bool lze přiřadit true, false, 1, 0 nebo booleovský výraz
- Konstanty nelze inicializovat ternárním operátorem
- Na index pole nelze přiřadit ternárním operátorem
- Pole a stringy nelze předávat do argumentu procedury
- Kód nacházející se mimo procedury může být pouze typu deklarace
- Je žádoucí, aby se veškeré deklarace děly na začátku souboru, resp. procedury
- Procedury musí být definovány dřív (výš v souboru), než budou volány
- Procedury nepodporují return.

3.2 Struktura projektu

Projekt je vedený na GitHubu (`https://github.com/EllaEstrellaM/FJP-C-to-PL0`). Práce je napsaná v jazyce Java a k tvorbě a průchodu stromu vytvořeného pomocí souboru gramatiky .g4 využívá ANTLR.

Projekt ve zdrojovém adresáři `src` obsahuje následující balíky (u výčtu viz stručný popis účelu):

- `compiler`
  stará se o samotnou kompilaci a generování instrukcí pro jednotlivé konstrukce jazyka, případně výpis informací o chybových stavech
  - `errors`
    stará se o výpis informací o chybách
  - `instructions_generators`
    obsahuje třídy, které řeší generování instrukcí pro jednotlivé konstrukce jazyka
- `generated`
  obsahuje třídy vygenerované pomocí ANTLR
- `statementDefMultiLine`
  obshajue víceřádkové definice konstrukcí jazyka, tj. třídy popisující cykly, if a definice procedur
- `statementDefOneLine`
  obshajue jednořádkové definice konstrukcí jazyka, tj. třídy popisující deklarace, přiřazení a volání procedur
- `statementInterEnum`
  obshajue výčtové typy a rozhraní definující strukturu projektu
- `visitors`
  obsahuje třídu typu `Visitor`, která je použitá k procházení vygenerovaného stromu

3.3 Generování struktur

Po spuštění programu jsou nejdříve vytvořeny struktury typů nacházejících se v balíku `statementDefMultiLine` a `statementDefOneLine` a jsou uloženy do seznamu. Tento seznam je následně předán třídě `Compiler`, která podle nich začne generovat instrukce.

3.4 Tabulky symbolů

Program si drží informace o deklarovaných proměnných a konstantách v tabulce symbolů. Existuje globální tabulka (obsahuje globální symboly) a také každá procedura má svoji privátní tabulku. Při kontrole existence symbolu se nejdříve kontroluje privátní tabulka, až poté globální, tzn. při deklaraci a následném použití lokálních proměnných (tj. uvnitř procedury) se program chová tak, jak bychom očekávali např. u C.

3.5 Volání procedur

Při volání procedury se kód volané procedury de facto dosadí do kódu volající procedury, dochází tedy k obdobě "inline expanzi".

3.6 Cyklus foreach

Cyklus foreach prochází jednotlivé prvky pole, které ukládá do proměnné specifikované v hlavičce cyklu. Ke své činnosti používá instrukci z rozšířené instrukční sady PL/0 - instrukci LDA.

4 Závěr

Z časových důvodů jsme nebyli schopni implementovat všechny konstrukce, které jsme původně implementovat chtěli, nicméně povinné konstrukce a výše jmenovaný seznam rozšíření jazyk umožňuje realizovat.