Uruchomienie

Program został napisany w Pythonie 3.10.6. Wykorzystywana jest konstrukcja `match: ... case:`, która jest dostępna od wersji 3.10. Użyłem biblioteki ANTLR4 w wersji 4.12.0

Program można uruchomić: `cat plik_zrodlowy.simple | python ./translator.py`

Wynik zostanie wysłany na `stdout`

Repozytorium jest dostępne pod adresem: https://github.com/mefju/aug

Uwagi:

Program drukuje tylko wartość wyrażeń arytmetycznych

Ponieważ język docelowy nie ma określonych wartości true/false, to uznałem, że wartości logicznych język źródłowy nie ma. Jeżeli trzeba, to mogę to zmienić.

Średnik po if

if_stat w wymaganiach ma na końcu definicji ';'. Zgodnie z przykładowym programem nie powinno, bo każda inna instrukcja kończy się ';'. Oznacza to, że zgodnie z definicją przykładowe 'if' .. 'then' powinno wyglądać: `if a>0 then a:=1;;` pierwszy średnik kończy przypisanie, a drugie if. Moja gramatyka, działa zgodnie z przykładowym programem.

Znak w wyrażeniach arytmetycznych

zgodnie z definicją języka pierwotnego znaki minus i plus mogą być tylko przed liczbami. To znaczy `a:=+1`, `a:=-1` są prawidłowymi przypisaniami, natomiast `a:=-a`, `a:=-(a+b)` nie.

Wartość początkowa zmiennych

W opisie języka nie jest zdefiniowana wartość początkowa zmiennych. Ponieważ w przykładzie pamięć została zainicjalizowana na 0, przyjąłem, że zmienne mają początkowo wartość 0. Z tego powodu:

- translator nie sprawdza, czy zmienna została użyta przed nadaniem jej wartości explicite przez użytkownika.
- w języku docelowym zmienne są inicjowane wartością 0.

Zdublowane deklaracje etykiet i zmiennych

Język przechodzi do porządku dziennego nad podwójnie zadeklarowanymi zmiennymi/etykietami: jeżeli coś zostało zadeklarowane dwa razy wewnątrz swojej

grupy (ale nie tu i tu!), to nie zatrzymuje to translacji programu.

Rzeczy do poprawy/udoskonalenia

Optymalizacja wyrażeń arytmetycznych i logicznych

Wykonywać obliczenia arytmetyczne jeżeli ich wynik jest znany. np.

```
a:=1+2;
```

jest zamieniane na:

```
0 PUSH 1
1 PUSH 2
2 POP 0
```

a mogłoby być na:

```
0 PUSH 3
1 POP 0
```

Podobnie sprawa ma się z wyrażeniami logicznymi, są one w całości wykonywane przez maszynę docelową.

Optymalizacja przypisań

Translator mógłby optymalizować przypisania. Na przykład:

```
a:=1;
a:=2;
a:=3;
```

zamienić na:

```
a:=3;
```

Instrukcja `NOP`

`NOP` jest wykorzystywana jako cel skoków. Możliwe jest usunięcie tej instrukcji i dokonywanie skoku bezpośrednio do instrukcji następnej

Komentarze

Translator mógłby opisywać w komentarzach języka docelowego co jest efektem działania translatora. Obecnie komentowane są jedynie skoki `goto`

Komunikaty błędów

W czasie translacji drukowane są niektóre komunikaty błędów, jeżeli źródłowy program zawiera błędy, ale duża część błędów programu źródłowego kończy się "gołymi" komunikatami błędów, bez np. podania numeru linii w której występuje