Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Michał Borkowski
Nr albumu: 370727

Jakub Bujak
Nr albumu: 370737

Marian Dziubiak
Nr albumu: 370784

Marek Puzyna
Nr albumu: 371359

Kompilacja NianioLanga do efektywnych struktur języka C

Praca licencjacka
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
mgr. Radosława Bartosiaka
Instytut TODO

Maj 2018

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpisy autorów pracy

Streszczenie

NianioLang jest językiem programowania ogólnego przeznaczenia, który można skompilować do wykonania na kilku platformach. Między innymi są to Java, JavaScript i C. Ze względu na chęć uproszczenia kompilacji NianioLanga utworzono środowisko uruchomieniowe przedstawiające odpowiednie abstrakcje, pozwalające na utworzenie dynamicznych struktur odpowiadających typom, jakie są dostępne do użycia w NianioLangu. Takie rozwiązanie nie jest niestety optymalne, szczególnie w przypadku niskopoziomowego języka jakim jest C. W tej pracy opisujemy wprowadzenie nowych typów i ich wsparcia w kompilatorze, które pozwolą na generowanie natywnego kodu w C, co znacznie zwiększy wydajność kompilowanych aplikacji.

Słowa kluczowe

kompilacja, języki programowania, analiza semantyczna, NianioLang, system typów

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

Klasyfikacja tematyczna

D. Software
D.3. Programming languages
D.3.3. Language contructs and features

Spis treści

Wprowadzenie

O tym że jest to praca licencjacka, o tym kto nam to zadanie zlecił, itp.

Rozdział 1

Wstęp

1.1. Czym jest NianioLang?

NianioLang jest proceduralnym, imperatywnym językiem, którego celem jest uproszczenie pisania rozproszonych aplikacji stosując wzorzec projektowy Nianio[1]. Celem twórców języka jest jego stosunkowa niskopoziomowość, pozwalająca na tworzenie szybkich aplikacji, również takich, które można uruchomić na urządzeniach o mniejszej mocy obliczeniowej czy też dostępnej pamięci. Jednocześnie konstrukcje takie jak wskaźniki zostały usunięte, a w ich miejsce zaimplementowano system zarządzania obiektami w pamięci przez zliczanie referencji, aby pozbyć sie jednego z najczęstszych źródeł błędów, jakie znajdują się w niskopoziomowych programach. Takie zabiegi zmniejszają nieco wydajność, ale twórcy języka doszli do wniosku, że zysk z uproszczenia wysokopoziomowego podejścia do pisania aplikacji jest wystarczająco duży, aby tworzenie aplikacji w NianioLangu było opłacalne.

Kompilator NianioLanga jest rozwijany przez firmę Atinea, która używa NianioLanga w swoich projektach (m.in. InstaDB). Kod źródłowy kompilatora jest dostępny na platformie GitHub na licencji MIT.

1.2. Typy w NianioLangu

W NianioLangu mamy doczynienia z kilkoma wbudowanymi typami. Wszelkie typy tworzone przez użytkownika, to właściwie aliasy na typy wbudowane, co pomaga w zarządzaniu abstrakcją w programie. Dostępne są cztery wbudowane typy:

- `ptd::sim` - który jest reprezentacją liczb całkowitych oraz ciągów znaków,
- `ptd::rec` - który reprezentuje słowniki/rekordy, gdzie do danego klucza mamy przypisaną pewną wartość,
- `ptd::var` - który określa typ wariantowy, czyli obiekt, który może być w dokładnie jednym z określonych stanów, i może dodatkowo zawierać dane, szczególne dla danego stanu,
- `ptd::array` - tablicę danych tego samego typu.

W wielu językach istnieje znacznie więcej wbudowanych typów, jednak powyższe są wystarczające, aby zbudować skomplikowane aplikacje, przy pomocy odpowiednich abstrakcji. W rozdziale *Kompilator NianioLanga* opisana została implementacja powyższych typów w języku C.

1.3. Cele projektu

Celem projektu jest modyfikacja kompilatora NianioLanga, w taki sposób, aby generowany kod w C zawierał mniejszą ilość wywołań funkcji oraz skomplikowanych struktur dla prostych typów istniejących już w języku C. Dodatkowo wprowadzone zostają nowe typy z przestrzeni nazw `own`, które będą bardziej niskopoziomowymi odpowiednikami typów z przestrzeni nazw `ptd` (mianowicie rekordy, tablice i warianty). Ich celem będzie znaczne zwiększenie wydajności programów pisanych w NianioLangu, które są kompilowane do C.

Rozdział 2

Metodyka pracy

2.1. Korzystanie z systemu kontroli wersji

2.2. Zgłaszanie zmian i code review

2.3. Techniki komunikacji w zespole

Rozdział 3

Kompilator NianioLanga

3.1. Budowanie drzewa AST

3.2. Analiza semantyczna

3.3. Architektura nlasma

Rejestry, wywołania funkcji, deklaracje typów, itp.

3.4. Translacja drzewa AST do nlasma

3.5. Generowanie kodu C na podstawie nlasma

3.6. Implementacja typów NianioLanga w C

Rozdział 4

Rozszerzenie systemu typów

4.1. Rozdzielenie typu `ptd::sim`

4.2. Typy `own`

Jaki jest cel tych typów, ich semantyka, jakie ograniczenia na ich użycie nakładamy (w stosunku do typów ptd).

Rozdział 5

Rozszerzenie nlasma

5.1. Przekazywanie informacji o typach z drzewa AST

5.2. Statyczne sprawdzanie poprawności

Rozdział 6

Nowe implementacje typów

W tej sekcji w każdym podrozdziale będziemy opisywać dlaczego dotychczasowe rozwiązanie było nieefektywne, jak można je było poprawić, które rozwiązanie wybraliśmy, dlaczego.

6.1. Typy proste

6.2. Tablice

6.3. Rekordy

6.4. Typy wariantowe

Rozdział 7

Efekty optymalizacji i wnioski

7.1. Porównanie czasu wykonania programów

Rozdział 8

Wkład własny

Co zrobiliśmy w rozbiciu na osoby

Bibliografia

[1] LK, *Wzorzec „Nianio" przykład* - 2006.