**Praca inżynierska**

**Karol Etrych**

kierunek studiów: **informatyka stosowana**

Opiekun: dr inż. Janusz Malinowski

**Kraków, styczeń 2017**

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i nie korzystałem ze źródeł innych niż wymienione w pracy.

.................................................................

(czytelny podpis)

Spis treści

[1. Wstęp 4](#_Toc501371140)

[1.1. język statycznie typowany, obiektowy, imperatywny. 4](#_Toc501371141)

[1.2. generujący kod zgodny ze specyfikacją ECMA-335 4](#_Toc501371142)

[1.3. spełniający wymogi zdefiniowane dla "Konsumenta CLI" przytoczyć + co się udało zrealizować, a co nie. 4](#_Toc501371143)

[1.4. Mechanizm wnioskowania typów. Przykłady w innych językach C#, F#, Scala. 4](#_Toc501371144)

[2. Parser 5](#_Toc501371145)

[1.1. AST 5](#_Toc501371146)

[1.2. Wybór sposobu implementacji parsera. 6](#_Toc501371147)

[1.3. Zasada działania parsera typu top-down. 6](#_Toc501371148)

[1.4. Problemy przy implementacji. 6](#_Toc501371149)

[3. Rezultaty pośrednie kompilacji 6](#_Toc501371150)

[1.1. Railway oriented programming 6](#_Toc501371151)

[4. Rozwiązywanie typów. 6](#_Toc501371152)

[1.1. Sprawdzanie czy typ jest zdefiniowany. 6](#_Toc501371153)

[1.2. Podmienianie specyfikatorów typu na w pełni kwalifikowane. 6](#_Toc501371154)

[5. Wnioskowanie typów. 6](#_Toc501371155)

[1.1. Algorytm działania. 6](#_Toc501371156)

[1.2. Algorytm znajdowania least-upper-bound. 6](#_Toc501371157)

[1.3. Dlaczego w AST pojawia się <'Expression> ? 6](#_Toc501371158)

[6. Sprawdzanie semantyki. 6](#_Toc501371159)

[1.1. Co jest sprawdzane. 6](#_Toc501371160)

[1.2. Reguły semantyki. 6](#_Toc501371161)

[7. Generowanie IR. 6](#_Toc501371162)

[8. Generowanie CIL. 6](#_Toc501371163)

[9. Użyte narzędzia i podsumowanie. 7](#_Toc501371164)

[1.1. F#, VSCode+Ionide, FParsec, Argu, Expecto, .NET Core 7](#_Toc501371165)

[10. Użycie kompilatora 7](#_Toc501371166)

[11. Bibliografia 7](#_Toc501371167)

# Wstęp

## język statycznie typowany, obiektowy, imperatywny.

## generujący kod zgodny ze specyfikacją ECMA-335

## spełniający wymogi zdefiniowane dla "Konsumenta CLI" przytoczyć + co się udało zrealizować, a co nie.

## Mechanizm wnioskowania typów. Przykłady w innych językach C#, F#, Scala.

# Parser

## AST



## Wybór sposobu implementacji parsera.

## Zasada działania parsera typu top-down.

## Problemy przy implementacji parsera.

# Struktura projektu

# Rezultaty pośrednie kompilacji (Railway oriented programming).

# Przetwarzanie AST. Katamorfizmy.

# Rozwiązywanie typów.

## Sprawdzanie czy typ jest zdefiniowany.

## Podmienianie specyfikatorów typu na w pełni kwalifikowane.

# Wnioskowanie typów.

## Algorytm działania.

Funkcje poniżej mogą bazować na typach funkcji zdefiniowanych w tym samym module wcześniej.

## Algorytm znajdowania least-upper-bound.

## Dlaczego w AST pojawia się <'Expression> ?

# Sprawdzanie semantyki.

## Co jest sprawdzane.

## Reguły semantyki.

1. Operandy operatorów binarnych muszą być tego samego typu.

2. Funkcje rekurencyjne muszą mieć podany zwracany typ.

# Generowanie reprezentacji pośredniej.

# Generowanie Intermediate Language.

# Użyte narzędzia i podsumowanie.

## F#, VSCode+Ionide, FParsec, Argu, Expecto, .NET Core

## Dalszy rozwój projektu.

W ramach dalszego rozwoju języka planowane jest dodanie nienulowalnych typów referencyjnych oraz wersja na .NET Standard (Linux).

# Użycie kompilatora

# Bibliografia