KONSTANTIN NISHT

Software developer

@ zeron.n1@yandex.ru +7 (904) 616-78-93 knisht
Saint Petersburg, Russia

EXPERIENCE

Software developer

JetBrains Research, HoTT team

July 2021 – Ongoing Saint Petersburg, Russia

- Supporting HoTT-based proof assistant Arend and the IntelliJ IDEA plugin for it.
- Implemented numerous refactorings and code insights for IntelliJ Arend. Enhanced AST in the language.

Software developer

JetBrains s.r.o., IntelliJ IDEA team

Sep 2019 – Ongoing Saint Petersburg, Russia

- Working in IntelliJ IDEA team, supporting plugin for Apache Groovy and Gradle buildscripts.
- Primarily focused on performance optimization of the flow typing and quality of type inference.

Software developer intern

JetBrains s.r.o.

Jul 2019 – Sep 2019 Saint Petersburg, Russia

- Implemented type inference algorithm for Groovy type system.
- Developed integration with Java type ecosystem tools. Improved existing instruments that aimed to support programming languages with dynamic type system.
- Presented project to IntelliJ IDEA team.

EDUCATION

B.Sc. Applied Mathematics and Computer Science

ITMO University

2017 – 2021 Saint Petersburg, Russia

- Computer Technology Department
- GPA: 4.95/5.0
- Relevant courses: Algorithms and Data structures, Type theory, Discrete Mathematics, Parallel programming, Operating systems, Computer architecture, C++ language, Advanced Java, Mathematical analysis, Probability theory and Mathematical Statistics.

ABOUT ME

I am a curious and hardworking person that is interested in computer science, programming languages and type theories.

PROGRAMMING LANGUAGES

Kotlin	●●●●●
Java	●●●●●
Arend	●●●●●
Haskell	●●●●○
Coq	●●●○○
Agda	●●●○○
C++	●●●○○

TECHNOLOGIES

Git · Linux · IntelliJ IDEA · Verified programming

LANGUAGES

Russian
Native

English
CEFR C1, IELTS 7.5/9

Chinese
Pre-intermediate

PROJECTS

- Term Rewriting Systems, formalization of this theory in Arend. The main result is a verified proof that a sum of linear confluential term rewriting systems is confluent itself.
- SLR(1) Parser Generator, written in Haskell.
- Ray tracing algorithm, implemented in Rust.