Object-oriented compiler design

Maroš Holko (xholko01@stud.fit.vutbr.cz)
Tomáš Kožár (xkozar02@stud.fit.vutbr.cz)

Abstract

The object-oriented paradigm is the most common paradigm in programming and it has been for quite some time. Some of its advantages are better system decomposition, readability, maintainability, and testability. Since compilers are complex systems with multiple parts, they can profit from proper object-oriented design.

This presentation will focus on designing the main compiler phases in an object-oriented way. Some parts become better testable as result of this design. Mainly scanner and parser. Symbol tables, while not being compiler phase directly, are an essential part of the compilation process and can be easily designed in object-oriented way as well as their contents, which can be designed into hierarchical classes.

Simulation of a derivation tree construction during syntax analysis can be designed in such a way that each node is an object. This allows us to tie these nodes to their semantic actions directly. Such object-oriented design also changes the way that code is generated in comparison to a non object-oriented imperative way. Each of these node objects knows how to generate their own code, which is then aggregated.