Pascal to JVM Compiler

Yao Du

2359451d

Proposal

Motivation

The project aims to implement a portable compiler for Pascal-like other trending compilers (Delphi, GNU Pascal, Free Pascal, .etc), which it could run on a series of platforms (with different combinations of operating systems and underlying architecture). The portability is achieved by introducing the concepts of Java bytecode as the intermediate code. It is known that Java is a platform-independent language. Since using the platform-dependent JVM (Java Virtual Machine) builds, we could run the bytecode generated by the Java compiler on all operating systems. Then JVM interprets the bytecode for generating the platform-specific machine code. Thus, we could run any bytecode generated by this project (portable Pascal compiler) on any platforms where the JVM is available.

Moreover, JVM is very mature and covers several tools for developers to do the arguments optimization and memory management, further promoting the performance of JVM-based programs.

Aims

Basic Behavior

The project should cover general constructs of Standard Pascal (ISO 7185) with minimum constructs of where users can run meaningful Pascal programs on installed JDK/JRE. Things like decision making, loops, procedures, functions, simple types, user-defined types, and using supported basic runtime built-ins: write, read, etc.

Assessment of the Compiler

Workability of the compiler (ability to handle how much part of the syntax and semantics of Standard Pascal) is measured using Regression Test (takes Custom Unit Test Suites as the basis). And might run Validation Test (operates existing Pascal Acceptance/Rejection Test Suites) on the compiler to detect the potential bugs in the compiler. Metrics might cover the test coverage or others if appropriate.

The test should be automated using scripts rather than testing manually, as that takes too long. Might complete the CI/CD workflows to trigger the testing and deployment (DockerHub). But if it takes too much time to arrange would leave for it. Anyway, introducing CI/CD would provide the project with good maintainability, even for a small individual project.

And the results of testing (covered any metrics) might be printed and gathered if possible. It is necessary to ensure the compiler passes the minimum test suites/cases given, guaranteeing that at least the project could be built and run as expected successfully on one platform (my hardware environment: Win os + x64 arch).

Extension

The project might cover concepts of other Pascal features, e.g., Object-Pascal, Extended-Pascal (ISO 10206), etc.

Three stages of the extension (if any) should work as expected.

Usage Extension

Might deploy the entire project to DockerHub, which saves time setting up all the dependencies required for this project using Docker.

The project might provide scripts to compile a program in a quick way with minimum behavior like `./script compile/run [path_to_program]` or more specific `./compile_script [path_to_program]` (i.e. with alias) rather than `java xxxEntrance compile/run [path_to_program]`.

A further extension might be a command-line tool that encapsulates the entire project without using the alias. So the user can use commands like `ptj compile [path_to_program]` to compile the program.

Progress

- Syntactic Analysis implemented using the automatic generating tool: Antlr(4.9.1). Both runtime lib, docker image, scripts are ready for regenerating lexer & parser if any
- Partial contextual analysis implemented for constructs covered: simple types (integer, real, char, boolean), vairable & procedure declarations, assignement & repetitive & if & procedure statements
- Set up test cases for syntactic analysis driver using Junit
- Set up Github Action CI/CD workflow only for building and deploying the Docker image where Antlr is ready to use with an alias (it still requires refined scripts to work).

Problems and risks

Problems

Some definitions of specific constructs are vague. It may not mention some concepts explicitly in the standard Pascal convention (ISO 7185). Online resources are confusing and do not mention whether a specific behavior is standard. Some constructs might not behave as expected by Standard Pascal due to referring to the documentation in a potentially problematic way.

Risks

As mentioned above, the current implementation has referred to various documentation of Pascal dialects (where one might not know whether the feature is truly standard) and standard one. **Mitigation:** would mainly focus on ISO 7185, avoiding introducing subjective understanding.

Also, it seems impossible to assess the "true concept" of **portability** without hardware. Ideally, as the compiler would generate bytecodes and JVM is portable to any platform supported, the compiler should work and be portable to anywhere with JVM installed. But without hardware, it is not sure whether the compiler could truly work and pass the test suites on different platforms. And to conduct the portability testing might take a long time. **Mitigation:** would introduce Docker to provide further portability. Prepare a ready runnable environment in a container, e.g., if the compiler works in Linux-x64, then set up an environment with this kernel.

- I'm not sure whether it makes sense to run the project in various virtual environments (say if it works in a virtual one, does it truly work in the actual hardware)

Plan

Winter semester

- **Week 1-6**: Setup GitHub repo and meeting pattern & Work on Syntactic Analysis
  - basic syntactic analysis was finished
- **Week 7-13**
  - partial contextual analysis was finished, but still got several constructs to be implemeted

Winter break

- Finalise the contextual analysis for minimum constructs
- Start working on code generation implementation and researching JVM things (mainly class files)
- Figure out a uniform way to write test cases (if it takes too much time, I will leave it for now) and how to make it automated (providing scripts, setup CI/CD, etc.)
  - If Junit is enough, would use it. Should also consider how to integrate the validation testing using existing test suites with regression testing
- Extend the project if code generation finished focus on implementing **other Pascal features** (OO feature etc.)
- if still got time, I might research more about making a **command-line tool** for usage.
- For dissertation preparation, should learn how to use bib and reference management. Otherwise, it seems problematic