Compilers Design Project Definition
Andres Quiroz Duarte

1. General Description of the Compiler

I decided to build a compiler for Cython, which is a hybrid from C and python and is a compiled language.
This lets you have the combined power of python and c or c++ at any point.
Python is a multi-paradigm language that covers object oriented, imperative programming and functional paradigm. It is an interpreted language, uses dynamic typed.
C is also a multi-paradigm language that covers imperative and procedural paradigms.It is a compiled language.
The reach of the project is explained later but the paradigms that are going to be implemented are: object-oriented(small part) and imperative paradigms on python and imperative paradigm on C.

2. Description of the Stages

For the lexical analysis every token needs to be classified in the assignation of variables or in the definition of a function.

For the Semantic Analysis I need to know the context of what the code is telling me so that the compiler knows if you are referring to one variable or another within the several scopes of loops and functions generate or give def the meaning as a keyword of starting a new function.

a. Lexical Analysis

As mentioned before, this is a project focused on Cython
This project will cover on both languages:

1. *Comments*.
2. *Nested structures*.
3. *Variables and Constants*.
4. *Strings*.
5. *Input and Output*.
6. *Data types*.
7. *Conditional and Loops*.
8. -Object-Oriented paradigm
9. -Imperative Paradigm

b. Syntax Analysis

Python has “def” as a keyword to start a function as in C there is no such keyword to start a function.
In C you have several keywords for data types such as “char” ,“int”,”float”,”boolean”, but in python, the language assumes the type of a variable by interpreting the assignation so there are no keywords for data type assignations.

c. Semantic Analysis

In this section, you need to briefly explain the semantic of your formal language. In the previous example, the meaning of `lambda` is used to create anonymous functions (i.e. functions that are not bound to a name) at runtime, however, it is not quite the same as lambda in functional programming languages.

d. Code Generation

In this section, you need to briefly explain what will be your code representation so it can be later executed. Here is when you decide if you will build a compiler (you will create an executable file), an interpreter (you will create an executable line of code) or a translator (ypu will create a new file in a different language).

3. References

Remember to include all the references you used.

1. https://en.wikipedia.org/wiki/Python_(programming_language)
2. https://www.tutorialspoint.com/python/python_basic_syntax.htm
3. https://en.wikipedia.org/wiki/Backus%E2%80%93Naur_form
4. https://docs.python.org/3/reference/grammar.html