Université Libre de Bruxelles

Introduction to language theory and compiling
INFO-F403

Project Pascalmaispresque
Part 1 - Scanner

Students :

Achten Alexandre 494484
Bienfait Alexandre 513930

Professors :

Geeraerts Gilles
Sassolas Mathieu
Brice Léonard

October 2023

Academic year 2023-2024

1 Lexical Analyzer

1.1 Regular expressions

VARNAME : [a-z][a-zA-z0-9]*
A VARNAME must be a string of letters and digits starting with a lower case letter. The corresponding regular expression ensure that a varname starts by a lowercase letter. It is followed by a Kleene closure of letters and digits

NUMBER : [0-9]+
A number is a string of digits. There must be at least one digit in it.

1.2 MACROS

WHITESPACE : [\t \r \n]+
The whitespaces have been defined as the return of line, space, tabulation and the carriage return.

1.3 States

The lexical analyzer can be in the following states.

There are 3 different states :

1. **Initial state**
   It is the basic state. It detects every key words of the language and return a symbol with the corresponding Lexical Unit. When encountering a character that isn't in the Lexical Unit, it raises an Illegal Character message.

2. **Long comment state**
   This state starts when there is a ' ' in the initial state. It ignores any other character except another ' ' that sends back to the initial state.

3. **Short comment state**
   This state start with the characters ** and end where there is a newline character (\n).

We could implement nested comments and recursively call the long comment state to "count" the depth of comments. But the difficulty is that in our language we identify long comments with " ' ' ". There is no difference between the beginning and the end of long comments.
To use nested comments, we would have to create a new character to distinguish the beginning and end of comments (e.g. "/*" and "*/").

1.4 Main file

The script in the main file takes as argument a file to analyse with the LexicalAnalyzer. It prints each token and it's lexical unit to the standard output. To print the symbol table, a `Hashmap` is used to store the line of the first occurrence of each variable. Then the symbol table is printed to the standard output.

2 Tests

Tests have been carried out to ensure that all functionalities are working. Here's a list of the different tests we've done :

- Test if spaces are not taken into account (regardless of the number of spaces)
- Test with an illegal character such as "A" or " ?"
- Test the recognition of variable names such as "whileif" or "alex83"
- Test the short and long comments
- Test a number followed by characters without spaces between them (the program don't work correctly for that case and just return a NUMBER followed by a VARNAME)
- Test if a NUMBER starts with 0 (we could change the definition of the regular expression NUMBER by [1-9][0-9]*, but we don't know if it's the purpose of the lexical analyzer)

To run all the tests, please use the `make tests` command.