

Assignment 1

Compilers II - CS3423

K S Ananth
CS22BTECH11029

Nimai Parsa
CS22BTECH11044

1 Conditional Statement

Alternative 2 is not context free, rather it is context sensitive. The number of indents for an if statement depends on the depth of the if statement in the nested **if-then-else** statement which makes it context sensitive. Therefore, alternative 1 is chosen over alternative 2.

2 Assumptions

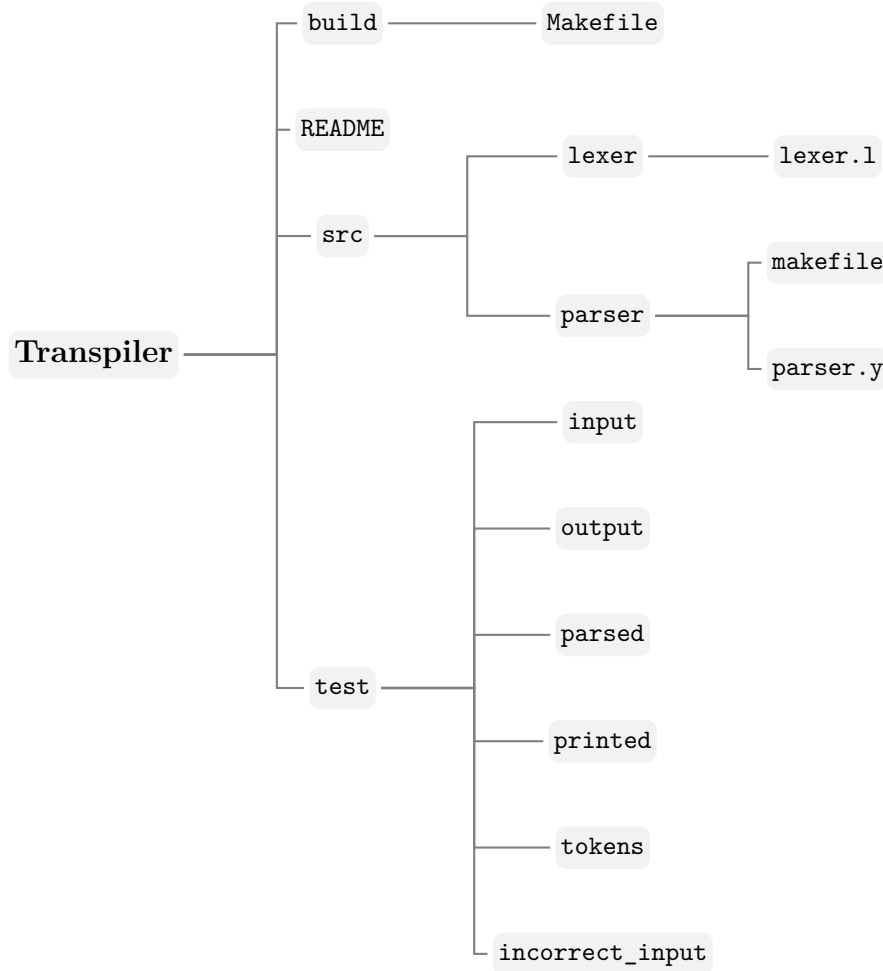
- All the statements (including set statements, unless it is a compound statement) are separated using semicolon (;).
- Print statement can only print int or float values.
- The print statement by default prints a new line ('\n') along with the value.
- '==' is used as the relational equality operator.
- Unary operators + and - cannot be repeated more than once in a row. Example ++i, +-i, -i all give syntax errors. Whereas, not+j, -not+j are valid syntax.
- Our transpiler doesn't support increment (++) and decrement (--) operators.
- Our transpiler supports assignment statement in the rhs of an assignment statement. Example, a = b = c; is not a syntax error.
- Our transpiler supports assignment statement in the access operator also. Example, a[b = 0] = 0; is not a syntax error.
- Our transpiler follows the following precedence and associativity order.

Operators	Associativity
not, +, -, size	Left
*, /, %	Left
+, -	Left
>, <, <=, >=	Left
==, <>	Left
&	Left
^	Left
	Left
and	Left
or	Left
=	Right

Table I: Operator Associativity

- Our predicate contains expressions separated by logical operators. Example, a * b >= c % 2 is valid syntax.
- Our transpiler assumes that the program body is not empty.

3 Steps to Compile and Run



The file structure of the Transpiler is shown above.

In order to run build and run this project follow the commands as listed below:

```
$ cd build
$ make
```

This will create all the required files according to the input files in the `./test/input` directory. It produces the following the transpiled cpp code in the `./test/output`, the result of the compiled and executed file in `./test/printed` directory, the tokens log file in the `./test/tokens` directory and the parser log file in the `./test/parsed` directory for the corresponding input file. It also runs the incorrect test cases present in the `./test/incorrect_input` directory and gives the syntax errors accordingly.

In order to clean the project run the following command:

```
$ make clean
```

The above command will remove all the files created by the makefile in build directory.

NOTE: The main.o object file after build will be present in both the build folder and the `./src/parser` directory as our build works in that way.

NOTE: Testing of files happens in lexicographical order.

NOTE: Please ignore any warnings that might pop up. The project works end to end perfectly.