Raffaele Tranquillini (ID: 17112)
Francesco Piccoli (ID: 17117)

Compilers Exam Project: Simple SCSS compiler

Project explanation

The project considers a subset of the *SCSS* language, an extension of *CSS* which allows the use of variables, functions and nested structures of declarations which are not originally present in *CSS*.

The compiler, generated using Lex, Yacc and C, first of all acts like a *linter* and is able to find errors in code related not only to the syntax of the language (as for example missing ";" or missing "{") but also to the semantics (i.e. sum between different unit measures "px" and "em" not allowed).

Declared variables are saved in symbol tables, since blocks might be nested (i.e. a structure of nested curly brackets), symbol tables are related in a tree-like fashion where each child block inherits all the declarations (attributes and variables) of the parent block and each block has its own symbol table. Each symbol table is implemented as a linked list where each record contains a pointer to the next record.

Installation instructions

To run the compiler, first of all *cd* to the "*SCSS_Compiler*" folder. At this point there are two possible ways to interact with the compiler: the first is by writing directly into the terminal the possible strings of the language, the other is by giving an input file via shell filters (already provided as "*test_wrong.scss*" and "*test_correct.scss*"):

1. `make; ./build/flc`
2. `make; ./build/flc ../test_correct.scss > out.css`
   OR: `make; cat ../test_wrong.scss | ./build/flc`

Input descriptions

Input should be similar to normal CSS code with the addition of variables (*$<name>: <value>*) and nested blocks, unsupported by normal CSS (e.g. *div { .col-md-5 { a { … }}}*). Here is an example:

```
$x : 20px;
$y: (20%)-2%;
/* comment */
.code {
       color: green;
       a {
              width: $y;
               li {
                  $z : $x * 2; } } }
```

Grammar

Scope: *S*

List of terminals/tokens:
{ID, NUM, UNIT, VAR, T_SEMICOLON(‘;’), T_COLON(‘:’), T_DOT(‘.’), T_COMMA(‘,’), T_HASH(‘#’), T_PL(‘(‘), T_PR(‘)’), T_BL(‘{‘), T_BR(‘}’), T_PLUS(‘+’), T_MINUS(‘-’), T_STAR(‘’), T_DIV(‘/’), T_GT(‘>’)}*

List of non-terminals:
{S, ST, VARDECL, EXPR, SCALAR, FNCALL, P, PARAMS, CSSRULE, SELECTORS, SELECTOR, DECLS, DECL}

S → ST S | *ε*
ST → VARDECL | CSSRULE
VARDECL→ VAR **:** EXPR **;**
EXPR → VAR | SCALAR | ID | COLORHEX | FNCALL | **(** EXPR **)** | EXPR **+** EXPR | EXPR **-** EXPR | EXPR ***** EXPR | EXPR **/** EXPR
COLORHEX → **#** ID | **#** NUM
SCALAR → NUM UNIT | NUM
FNCALL → ID **(** P **)**
P → EXPR PARAMS | *ε*
PARAMS → **,** EXPR PARAMS | *ε*
CSSRULE → SELECTORS **{** DECLS **}**
SELECTORS → SELECTOR PSEUDOCLASS RELATIONSHIP
SELECTOR → ID | **#** ID | **.** ID
PSEUDOCLASS → **:** ID | *ε*
RELATIONSHIP → **,** SELECTORS | **>** SELECTORS | SELECTORS | *ε*
DECLS → DECL DECLS | *ε*
DECL → ID **:** EXPR **;** | CSSRULE | VARDECL