Imperial College London

Language Processors

Lab 4 – Lexical analysis with `flex`

Max Cattafi (m.cattafi@imperial.ac.uk)

`flex` (an open and free version of `lex`) is a tool which takes in input `.l` configuration files and generates accordingly C code for programs (called "lexers" or "scanners") that perform on text given in input pattern-matching associated to actions.

Files `simple1a-c.l`, `simple1a-cpp.l` and `simple1b.l` contain (commented and explained) examples of `.l` files.

Generate a scanner:

```
flex -o simple1a-c.c simple1a-c.l
```

The resulting C source file can be compiled as usual:

```
gcc simple1a-c.c -o simple1a-c
```

It is also possible to generate and compile the scanner in one line using the shell `&&` operator:

```
flex -o simple1a-c.c simple1a-c.l && gcc simple1a-c.c -o simple1a-c
```

By default scanners read and write from and to the standard I/O. Using `cat`, the pipe `|` and redirection `>` we can get input from file `idnum.txt` and write the output on file `1aout.txt`:

```
cat idnum.txt | ./simple1a-c > 1aout.txt
```

Although `flex` generates scanners in C, it is possible to include C++ code in the scripts as long as the scanner is generated as a `.cpp` file and compiled accordingly (in this case the C code generated by `flex` is mixed with the C++

code in the script and it is possible to compile it all as a C++ program). For instance:

```
flex -o simple1a-cpp.cpp simple1a-cpp.l && g++ simple1a-cpp.cpp -o simple1a-cpp
```

Generate and compile scanners from the given scripts and test them using file `idnum.txt`.

Write a `flex` script for a scanner which, given in input some source file like `prog2.cpp` outputs a version of the file where recognized tokens are replaced by the token label (and, in some cases, the token attribute in angular parentheses): see `prog2out.txt`.