Week 5

Exercise 35

../35/flexcpp/lexer

```
%%
[a-zA-Z]+  return matched()[0];
[ \t\n]    // Ignored;
```

../35/flexcpp/Scanner.h

```
// Generated by Flexc++ V2.06.02 on Fri, 08 Mar 2019 01:22:12 +0100

#ifndef Scanner_H_INCLUDED_
#define Scanner_H_INCLUDED_

// $insert baseclass_h
#include "Scannerbase.h"


// $insert classHead
class Scanner: public ScannerBase
{
    public:
        explicit Scanner(std::istream &in = std::cin,
                         std::ostream &out = std::cout);

        Scanner(std::string const &infile, std::string const &outfile);

        // $insert lexFunctionDecl
        int lex();

    private:
        int lex__();
        int executeAction__(size_t ruleNr);

        void print();
        void preCode();     // re-implement this function for code that must
                            // be exec'ed before the patternmatching starts

        void postCode(PostEnum__ type);
                            // re-implement this function for code that must
                            // be exec'ed after the rules's actions.
};

// $insert scannerConstructors
inline Scanner::Scanner(std::istream &in, std::ostream &out)
:
    ScannerBase(in, out)
{}

inline Scanner::Scanner(std::string const &infile, std::string const &outfile)
:
    ScannerBase(infile, outfile)
{}

// $insert inlineLexFunction
inline int Scanner::lex()
{
    return lex__();
}

inline void Scanner::preCode()
{
```

```
54      // optionally replace by your own code
55  }
56
57  inline void Scanner::postCode(PostEnum__ type)
58  {
59      // optionally replace by your own code
60  }
61
62  inline void Scanner::print()
63  {
64      print__();
65  }
66
67
68  #endif // Scanner_H_INCLUDED_
```

../35/main.cc

```
1   #include "main.ih"
2
3   int main(int argc, char **argv)
4   try
5   {
6     multiset<string> words = (argc != 1) ? processfiles(argc, argv) : processcin();
7     // Multiset to order input
8
9     for (auto el: words)    // Print output
10      cout << el << '\n';
11  }
12  catch (string &message)
13  {
14    cout << message;
15  }
```

../35/fileexists.cc

```
1   #include "main.ih"
2
3   bool file_exists(const char *fileName)
4   {
5     ifstream infile(fileName);
6     return infile.good();
7   }
```

../35/processcin.cc

```
1   #include "main.ih"
2
3   multiset<string> processcin()
4   {
5     multiset<string> words;
6
7     Scanner flatbed;  // Default constructor (cin, cout)
8
9     while (flatbed.lex())
10      words.insert(flatbed.matched());
11
12    return words;
13  }
```

../35/processfiles.cc

```
1   #include "main.ih"
2
```

```
3  multiset<string> processfiles(int argc, char **argv)
4  {
5    multiset<string> words;
6
7    for (int idx = 1; idx != argc; ++idx) // Loop through files
8    {
9      if (!file_exists(argv[idx]))        // Check if input file(s) exist
10     {
11       string message = argv[idx];
12       message += " is not a valid file.";
13       throw message;
14     }
15
16     Scanner flatbed(argv[idx], "/dev/null");  // Scanner from filename
17     // While not elegant, writing the output to /dev/null is the simplest
18     // method I could think of, considering the fact that it should,
19     // indeed, not be stored. Another option is to create a custom constructor,
20     // or associating a filebuf with an istream and passing that to a Scanner
21     // constructor (using cout as the other argument, since it will not output
22     // anything). However, this seemed both the cleanest and clearest option.
23
24     while (flatbed.lex())                    // While input
25     {
26       words.insert(flatbed.matched());  // Insert next match into multiset
27     }
28   }
29   return words;
30 }
```

Exercise 36

../36/flexcpp/lexer

```
operators       [\/\+\-\*\%\=\!\&\|\~\^\<\>]
%%

[ \t\n]         // Ignored
([a-zA-Z]{2,})  return WORD;
[a-zA-Z]        return matched()[0];
[0-9]+\.[0-9]+  return FLOAT;
[0-9]+          return INT;
\".*            return STRING;
{operators}+    return OP;
```

../36/flexcpp/Scanner.h

```
// Generated by Flexc++ V2.06.02 on Sat, 09 Mar 2019 02:25:30 +0100

#ifndef Scanner_H_INCLUDED_
#define Scanner_H_INCLUDED_

enum Tokens
{
  INT = 256,
  FLOAT,
  STRING,
  WORD,
  OP
};

// $insert baseclass_h
#include "Scannerbase.h"


// $insert classHead
class Scanner: public ScannerBase
{
    public:
        explicit Scanner(std::istream &in = std::cin,
                         std::ostream &out = std::cout);

        Scanner(std::string const &infile, std::string const &outfile);

        // $insert lexFunctionDecl
        int lex();

    private:
        int lex__();
        int executeAction__(size_t ruleNr);

        void print();
        void preCode();     // re-implement this function for code that must
                            // be exec'ed before the patternmatching starts

        void postCode(PostEnum__ type);
                            // re-implement this function for code that must
                            // be exec'ed after the rules's actions.
};

// $insert scannerConstructors
inline Scanner::Scanner(std::istream &in, std::ostream &out)
:
    ScannerBase(in, out)
{}
```

```
49
50  inline Scanner::Scanner(std::string const &infile, std::string const &outfile)
51  :
52      ScannerBase(infile, outfile)
53  {}
54
55  // $insert inlineLexFunction
56  inline int Scanner::lex()
57  {
58      return lex__();
59  }
60
61  inline void Scanner::preCode()
62  {
63      // optionally replace by your own code
64  }
65
66  inline void Scanner::postCode(PostEnum__ type)
67  {
68      // optionally replace by your own code
69  }
70
71  inline void Scanner::print()
72  {
73      print__();
74  }
75
76
77  #endif // Scanner_H_INCLUDED_
```

../36/main.cc

```
1  #include "main.ih"
2
3  int main(int argc, char const **argv)
4  {
5    if (!file_exists(argv[1]))
6    {
7      cout << "Invalid file.";
8      return 1;
9    }
10
11   return processfile(argv[1]);
12 }
```

The same as in exercise 35:

../36/fileexists.cc

```
1  #include "main.ih"
2
3  bool file_exists(const char *fileName)
4  {
5    ifstream infile(fileName);
6    return infile.good();
7  }
```

../36/processfile.cc

```
1  #include "main.ih"
2
3  int processfile(const char *file)
4  {
5    size_t curLine = 0;
6    Scanner flatbed(file, "/dev/null");
7    while (true)
```

```
8     {
9       if (curLine != flatbed.lineNr())
10      {
11        curLine = flatbed.lineNr();
12        cout << "\nLine " << curLine << ": ";
13      }
14      switch (flatbed.lex())
15      {
16        case 0:
17          return 0;
18        case INT:
19          cout << "INT: " << flatbed.matched() << ' ';
20          break;
21        case WORD:
22          cout << "WORD: " << flatbed.matched() << ' ';
23          break;
24        case FLOAT:
25          cout << "FLOAT: " << flatbed.matched() << ' ';
26          break;
27        case STRING:
28          cout << "STRING: " << flatbed.matched() << ' ';
29          break;
30        case OP:
31          cout << "OPERATOR: " << flatbed.matched() << ' ';
32          break;
33        default:
34          cout << "CHAR: " << flatbed.matched() << ' ';
35          break;
36      }
37    }
38  }
```

Exercise 37

../37/lexer

```
1
2  %x multiline
3  %x hash
4  %x ccomment
5  %x string
6
7  %%
8
9  "\""                        begin(StartCondition__::string);
10 <string>["][ ]*             begin(StartCondition__::INITIAL);
11 <string>.|\n                echo();
12
13 "/*"                        begin(StartCondition__::multiline);
14 <multiline>"*/"[ ]*         begin(StartCondition__::INITIAL);
15 <multiline>.|\n
16
17 "#"                         begin(StartCondition__::hash);
18 <hash>"\n"                   begin(StartCondition__::INITIAL);
19 <hash>.|\n
20
21 //ccomment miniscanner beter leesbaar dan normale lexer code
22 "//"                        begin(StartCondition__::ccomment);
23 <ccomment>"\n"              begin(StartCondition__::INITIAL);
24 <ccomment>.|\n
25
26
27 ^\n
28 ^(\ )+
29 ^(\t)+
```

../37/Scanner.h

```
1  // Generated by Flexc++ V2.06.02 on Fri, 15 Mar 2019 16:14:20 +0100
2
3  #ifndef Scanner_H_INCLUDED_
4  #define Scanner_H_INCLUDED_
5
6  enum Token
7  {
8    MULTILINE = 257,
9    HASH,
10 };
11 // $insert baseclass_h
12 #include "Scannerbase.h"
13
14
15 // $insert classHead
16 class Scanner: public ScannerBase
17 {
18     public:
19         explicit Scanner(std::istream &in = std::cin,
20                          std::ostream &out = std::cout);
21
22         Scanner(std::string const &infile, std::string const &outfile);
23
24         // $insert lexFunctionDecl
25         int lex();
26
27     private:
28         int lex__();
29         int executeAction__(size_t ruleNr);
```

```
30
31          void print();
32          void preCode();     // re-implement this function for code that must
33                              // be exec'ed before the patternmatching starts
34
35          void postCode(PostEnum__ type);
36                              // re-implement this function for code that must
37                              // be exec'ed after the rules's actions.
38  };
39
40  // $insert scannerConstructors
41  inline Scanner::Scanner(std::istream &in, std::ostream &out)
42  :
43      ScannerBase(in, out)
44  {}
45
46  inline Scanner::Scanner(std::string const &infile, std::string const &outfile)
47  :
48      ScannerBase(infile, outfile)
49  {}
50
51  // $insert inlineLexFunction
52  inline int Scanner::lex()
53  {
54      return lex__();
55  }
56
57  inline void Scanner::preCode()
58  {
59      // optionally replace by your own code
60  }
61
62  inline void Scanner::postCode(PostEnum__ type)
63  {
64      // optionally replace by your own code
65  }
66
67  inline void Scanner::print()
68  {
69      print__();
70  }
71
72
73  #endif // Scanner_H_INCLUDED_
```

../37/main.cc

```
1  #include "main.ih"
2  #include "Scanner.h"
3
4  int main(int argc, char const **argv)
5  {
6    Scanner hp("input", "output");
7    hp.lex();
8  }
```

Exercise 38

../38/lexer

```
%x string


%%

"\""                        {
                              begin(StartCondition__::string);
                              setMatched("grabbed(" + counter() + ", " + \
                              filename() + ".gsl);");
                              echo();
                            }
<string>["][;]              {
                              begin(StartCondition__::INITIAL);
                            }
<string>.|\n
```

../38/Scanner.h

```
// Generated by Flexc++ V2.06.02 on Fri, 15 Mar 2019 17:38:07 +0100

#ifndef Scanner_H_INCLUDED_
#define Scanner_H_INCLUDED_

// $insert baseclass_h
#include "Scannerbase.h"
#include <string>


// $insert classHead
class Scanner: public ScannerBase
{
        size_t d_counter = 0;

    public:
        explicit Scanner(std::istream &in = std::cin,
                                 std::ostream &out = std::cout);

        Scanner(std::string const &infile);
        Scanner(std::string const &infile, std::string const &outfile);

        // $insert lexFunctionDecl
        int lex();

        std::string counter();

    private:
        int lex__();
        int executeAction__(size_t ruleNr);

        void print();
        void preCode();     // re-implement this function for code that must
                            // be exec'ed before the patternmatching starts

        void postCode(PostEnum__ type);
                            // re-implement this function for code that must
                            // be exec'ed after the rules's actions.
};

// $insert scannerConstructors
inline Scanner::Scanner(std::istream &in, std::ostream &out)
:
```

```
44      ScannerBase(in, out)
45  {}
46
47  inline Scanner::Scanner(std::string const &infile)
48  :   ScannerBase(infile, infile + ".tmp")
49  {}
50
51
52  inline Scanner::Scanner(std::string const &infile, std::string const &outfile)
53  :
54      ScannerBase(infile, outfile)
55  {}
56
57  inline std::string Scanner::counter()
58  {
59    return std::to_string(++d_counter);
60  }
61
62  // $insert inlineLexFunction
63  inline int Scanner::lex()
64  {
65      return lex__();
66  }
67
68  inline void Scanner::preCode()
69  {
70      // optionally replace by your own code
71  }
72
73  inline void Scanner::postCode(PostEnum__ type)
74  {
75      // optionally replace by your own code
76  }
77
78  inline void Scanner::print()
79  {
80      print__();
81  }
82
83
84  #endif // Scanner_H_INCLUDED_
```

../38/main.cc

```
1  #include "main.ih"
2
3  int main(int argc, char const **argv)
4  {
5    if (argc == 2)
6    {
7      Scanner scanner(argv[1]);
8      scanner.lex();
9    }
10   else
11   {
12     cout << "Please provide input filename.\n";
13   }
14 }
```