C++ Exercises

Set 6 resend

Author(s): Pau Lopez, Peter Versluis

17:24

April 1, 2025

49

A grammar that parses function calls

Listing 1: grammar

```
%token WRITE IDENT NUMBER

%%

multiple_calls:
    // no call at all
|
    multiple_calls call
;

call:
    WRITE '(' arg_list ')'';'
;

arg_list:
    arg
|
    arg_list
    ','
    arg
;

arg:
    IDENT
|
    NUMBER
;
```

55

An interactive program that reads different commands.

Listing 2: lexer

```
%interactive
%target-directory scanner

%%

[ \t]                       // skip white spaces

'-'?[0-9]+\.?[0-9]*         |
'-'?\.[0-9]+                return Parser::DOUBLE;

'\n'                        return Parser::ENDL;

.                           return matched().front();
```

Listing 3: grammar

```
%scanner ../scanner/Scanner.h
%target-directory parser

%token ENDL
%token DOUBLE

%%

lines:
    empty_line
|
    full_line lines
;

empty_line:
    prompt ENDL
    {
        ACCEPT();
    }
;

full_line:
    prompt command ENDL
;

command:
    'm' DOUBLE
|
    's' DOUBLE
|
    DOUBLE
|
    DOUBLE DOUBLE   // integral values are also doubles
;

prompt:
{
    std::cout << "? ";
}
;
```

Listing 4: sol2/grammar

```
// the lexer file is the same for the 2nd solution.

%scanner ../scanner/Scanner.h
%target-directory parser

%token ENDL
%token DOUBLE

%%

lines:
    empty_line
|
    full_line lines
;

empty_line:
    ENDL
    {
        ACCEPT();
    }
;

full_line:
    command ENDL
    {
        d_newLine = true;
```

```
    }
;

command:
    'm' DOUBLE
|
    's' DOUBLE
|
    DOUBLE
|
    DOUBLE DOUBLE   // integral values are also doubles
;
```

Listing 5: sol2/Parser.h

```
// Generated by Bisonc++ V6.08.00 on Tue, 01 Apr 2025 12:49:44 +0200

#ifndef Parser_h_included
#define Parser_h_included

// $insert baseclass
#include "Parserbase.h"
// $insert scanner.h
#include "../scanner/Scanner.h"


// $insert undefparser
#undef Parser
    // CAVEAT: between the baseclass-include directive and the
    // #undef directive in the previous line references to Parser
    // are read as ParserBase.
    // If you need to include additional headers in this file
    // you should do so after these comment-lines.


class Parser: public ParserBase
{
    // $insert scannerobject
    Scanner d_scanner;
    bool d_newLine = true;

    public:
        Parser() = default;
        int parse();

    private:
        void error();                   // called on (syntax) errors
        int lex();                      // returns the next token from the
                                        // lexical scanner.
        void print();                   // use, e.g., d_token, d_loc
        void exceptionHandler(std::exception const &exc);

    // support functions for parse():
        void executeAction_(int ruleNr);
        void errorRecovery_();
        void nextCycle_();
        void nextToken_();
        void print_();
};


#endif
```

Listing 6: sol2/Parser.ih

```
#include "Parser.h"


inline void Parser::error()
{
    std::cerr << "Syntax error\n";
```

```
}

// $insert lex
inline int Parser::lex()
{
    if (d_newLine)
    {
        std::cout << "? ";
        d_newLine = false;
    }

    return d_scanner.lex();
}

inline void Parser::print()
{
}

inline void Parser::exceptionHandler(std::exception const &exc)
{
    throw;                  // re-implement to handle exceptions thrown by actions
}
```

?? (WA?) VC for inline. NSC: why re-implement it?