The University of Western Australia
Department of Electrical & Electronic Engineering
A/Prof Thomas Bräunl

# Programming Languages and Software Design
PLSD 210

## Lab Assignment 1 Due: week 3

Complete the following programming assignment using either
- the “gnu” C or C++ compiler “gcc” together with the “DOC++” documenting tool, or
- the Java JKD tool kit

Write a program that evaluates an input string representing a Lisp expression. This can be either a **single integer** or a possible **nested expression in parenthesis**: **(**operator op1 op2**)**
- Operators are **+**, **–**, *****, and **/**.
- Operands are positive integer numbers or Lisp expressions (*recursion*).

| | | | |
|---|---|---|---|
| **Examples:** | 7 | evaluates to | 7 |
| | (+ 1 4) | evaluates to | 5 |
| | (* (+ 2 3) (– 5 3)) | evaluates to | 10 |
| | (+ (1 2) (+3 4)) | evaluates to | error |

**Note:**
• Be careful and use some time for program design before starting coding.
• Document all procedures right from the start of coding using DOC++.
• Make sure your program reads its input data from “stdin” !

**Simplifying Assumption:**
You may assume that the input string contains no other symbols than:
(,),+,–,*,/,0,1,2,3,4,5,6,7,8,9, <space>
You may assume that only single digit numbers are allowed.

**Bonus Point:** For extra credit, allow the input string to contain “white space” (blanks, tabs, newline/carriage returns) at arbitrary positions and allow regular integer constants, not limited to 0–9. The latter part requires programming of a short “scanner” routine.

**Submit:** Submission is electronic and automatic. Create a subdirectory “**submit**” in your home area and place the source code only of your assignment in it. At **Fri. 8pm** of the due date of each assignment, the contents of all “submit” directories will be forwarded automatically.

The University of Western Australia
Department of Electrical & Electronic Engineering
A/Prof Thomas Bräunl

# Programming Languages and Software Design
PLSD 210

## Lab Assignment 2 Due: week 5

Use the following tools to complete this assignment:
- C or C++ compiler gcc with DOC++ for documentation, or
- Java JDK

Consider the groups of symbols:

| | | |
|---|---|---|
| word symbols | : | PROGRAM, VAR, BEGIN, END, TIMES, DO, STRAIGHT, LEFT, RIGHT (all in **uppercase** letters) |
| identifier | : | a name in **lowercase** letters (max 10 characters long) |
| constant | : | integer constant (max. 5 digits long) |
| special symbols | : | “,” , “**;**” , “=” |

Sample input:
```
PROGRAM VAR count; BEGIN count=4; TIMES count DO STRAIGHT; RIGHT END END
```

**Implement a scanner for Draw.**
The input text has to be digested into keywords, identifiers, constants, and special symbols.
E.g. the sample program becomes the token sequence:
<prog_sym> <var_sym> <identifier> <semicolon_sym> <begin_sym>
<identifier> <equal_sym> <constant> <semicolon_sym>
<times_sym> <identifier> <do_sym> <straight_sym> <semicolon_sym> <right_sym> <end_sym>
<end_sym>

**Hints**
1. Implement a function to return the current input character `char CurChar();`
2. Use a function to advance one character `char NextChar();`

The University of Western Australia
Department of Electrical & Electronic Engineering
A/Prof Thomas Bräunl

# Programming Languages and Software Design
PLSD 210

## Lab Assignment 3 — Due: week 7

Use the following tools to complete this assignment:

- C or C++ compiler gcc with DOC++ for documentation, or
- Java JDK

Consider the programming language “Draw”, defined in EBNF:

```
program      =   PROGRAM  declaration  BEGIN  stateseq  END .
declaration  =   [ VAR identifier { “,” identifier } “;” ] .
stateseq     =   statement { “;” statement } .
statement    =   [ identifier “=” expression |
                   TIMES identifier DO stateseq END |
                   STRAIGHT | LEFT | RIGHT ] .
expression   =   constant | identifier .
constant     =   digit {digit } .          (* max. 5 *)
identifier   =   letter { letter } .       (* max. 10 *)
letter       =   “a” .. “z” .
```

Sample Draw program:

```
PROGRAM
VAR count;
BEGIN
  count = 4;
  TIMES count DO STRAIGHT; RIGHT END
END
```

**Implement a recursive descent parser for Draw.**
The parser calls the scanner and checks the syntactical correctness of the program.
Error are to be reported – parsing stops at the first error encountered.

**Hints**

1. Implement a function to return the current input symbol `symboltype CurSymbol();`

2. Use scanner from previous lab to advance one symbol `symboltype NextSymbol();`

3. Use function “error”:
```
void error(char *s)
{ printf(“ERROR: %s\n”, s); /* print error message */
  myexit();                 /* terminate program */
}
```

4. Use function “check” to see if a specific keyword is present:
```
bool check(symboltype s)
{ if (s == CurSymbol())
  { NextSymbol(); return TRUE; }      /* symbol is present */
  else return FALSE;                  /* symbol is not there */
}
```

5. Use function “accept” if a specific keyword has to follow:
```
void accept(symboltype s)
{ if (!check(s)) error(“wrong symbol”);  /* must accept */
}
```

**Bonus Point:** Add arithmetic expressions to the language Draw.
Syntax change:

```
expression =   (constant | identifier)  [ operation  (constant | identifier) ] .
operation  =   “+”  |  “-”  |  “*”  |  “/” .
```

Remember to extend the scanner for symbols “+” , “-” , “*” , “/”

The University of Western Australia
Department of Electrical & Electronic Engineering
A/Prof Thomas Bräunl

# Programming Languages and Software Design
PLSD 210

## Lab Assignment 4 Due: week 9

Use the following tools to complete this assignment:
- C or C++ compiler gcc with DOC++ for documentation, or
- Java JDK

Sample Draw program:
```
PROGRAM
VAR count;
BEGIN
  count = 4;
  TIMES count DO STRAIGHT; RIGHT END
END
```

**Implement an interpreter for Draw.**

Extend the previously built scanner and parser by constructing a symbol table for identifier names together with their values. Now, also check for semantics errors, i.e. in statements report the use of identifiers, which have not been declared previously.

Extend the compiler with an interpreter by generating "**d-code**".
For each command, a single character is generated: STRAIGHT becomes "S", LEFT becomes "L", and RIGHT becomes "R". Each TIMES-loop has to be unrolled by repeating the string generated inside the loop for the specified number of times.

So the d-code generated from the above sample program is:

```
SRSRSRSR
```

**Bonus Point:** Implement the expression extension of the previous assignment.

The University of Western Australia
Department of Electrical & Electronic Engineering
A/Prof Thomas Bräunl

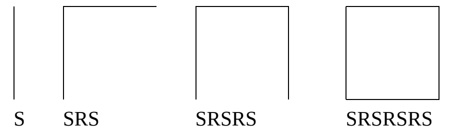
# Programming Languages and Software Design
PLSD 210

## Lab Assignment 5 Due: week 11

Use the following tools to complete this assignment:
- C or C++ compiler gcc with DOC++ for documentation **and** FORMS for graphics interface, or
- Java JDK

Sample d-code: Corresponding graphics (stepwise):

```
SRSRSRSR
```



**Implement a graphics code interpreter for *d-code*.**

Use C/C++ plus the FORMS graphics library or Java to implement a graphics code interpreter. Read each character sequentially from the generated file and perform the appropriate drawing command in "turtle graphics style". That is, start in the middle of the screen with the "turtle" pointing up. For each "S" code, draw a line of unit length in the current direction. The line's end-point becomes the new starting point. For codes "L" and "R" change the current direction 90 degrees to the left or right, respectively. The result of the sample program above is a square.

- Use a file-selector-box to read the input string.
- Incorporate slide bars for setting line length and turning angle.

**Bonus Point:** Generate a complete "programming environment". This includes file selection, compilation with d-code generation, and subsequent drawing on the click of one or more buttons.

The University of Western Australia
Department of Electrical & Electronic Engineering
A/Prof Thomas Bräunl

# Programming Languages and Software Design

PLSD 210

Lab Assignment 6 Due: week 13

Use the following tools to complete this assignment:

- C or C++ compiler gcc with DOC++ for documentation or
- Java JDK

**Implement a cross compiler / code generator to an M68000 Assembler system for Draw.** Instead of interpreting the "d-code" program code, generate an assembly language program (for Motorola 68000) to execute the program on a different platform, here the "**EyeBot**" controller.

The generated program should draw the graphics on the screen. Please note that the graphics screen runs from (x,y = 0,0) in the top left corner to (x,y = 127,63) in the bottom right corner. Use the following graphics function and push the parameters on the stack from right to left.

```
void LCDLine( int x1, int y1, int x2, int y2, int col)
/* draws line from (x1,y1) to (x2,y2) in color white (0), black (1) or inverse (2) */
```

- start at position (x,y = 30, 50) with orientation up (towards lower y-values)
- S should draw a line of fixed length 10 pixels
- L,R should change the orientation by +/- 90 degrees

**Hints**

- Write subroutines for **S**, **L**, and **R** - and **I** for init. - which are included in the generated code
- For debugging purposes, start with writing the characters "S", "L", and "R" on the screen using `printf()`. When this is working, proceed with the graphics functions.
- You may restrict the application (with appropriate error message) to allow a maximum of 4 nested loops.
- Further documentation: `http://www.ee.uwa.edu.au/~braunl/eyebot`

**Example for simple program generated**

```
main:   JSR I
        MOVE.L #4, D1
loop1:  DBRA D1, end1
        JSR S
        JSR R
end1:   RTS
```

**Bonus Point:** Implement the expression extension of the earlier assignment.