G++

G++ is a language being developed for teaching purposes at Gebze Technical University. This language has the following "vision":

- Lisp like syntax
- Interpreted
- Imperative, non-object oriented
- Static scope, static binding, strongly typed, …
- A few built-in types to promote exact arithmetic for various domains such as computational geometry

1

G++ Interpreter

- Starting G++ without an input file…

  $ g++

  > _ \\READ-EVAL-PRINT loop starts here…

- Starting G++ with an input file…

  $ g++ myhelloword.g++

  \\READ-EVAL-PRINT everything in the file…

  > _ \\READ-EVAL-PRINT loop starts here…

2

G++ – Lexical Syntax

- Keywords: *and, or, not, equal, less, nil, list, append, concat, set, def, for, if, exit, load, display, true, false*
- Operators: *+ - / * ( ) ,*
- Comment: Line or part of the line starting with *;;*
- Terminals:
  - *Keywords*
  - *Operators*
  - *Literals: There is only predefined type in this language.*
    - *Unsigned fractions – two unsigned integers separated by the character "b". E.g., 123b12 is the fraction* $\frac{123}{12}$
  - *Identifier: Any combination of alphabetical characters and digits with only leading alphabetical characters.*

3

G++ Lexer Tokens

KW_AND, KW_OR, KW_ NOT, KW_EQUAL, KW_LESS, KW_NIL, KW_LIST, KW_APPEND, KW_CONCAT, KW_SET, KW_DEF, KW_FOR, KW_IF, KW_EXIT, KW_LOAD, KW_DISPLAY, KW_TRUE, KW_FALSE

OP_PLUS, OP_MINUS, OP_DIV, OP_MULT, OP_OP, OP_CP, OP_COMMA

COMMENT

VALUEF

IDENTIFIER

4

G++ – Concrete Syntax

- Non-terminals:
  - $START, $INPUT, $EXP, $FUNCTION

5

G++ – Concrete Syntax

- $START -> $EXP | $FUNCTION | OP_OP KW_EXIT OP_CP

6

G++ – Concrete Syntax

- An expression always returns a fraction
- Expressions:
  - $EXP -> OP_OP OP_PLUS $EXP $EXP OP_CP |
    OP_OP OP_MINUS $EXP $EXP OP_CP |
    OP_OP OP_MULT $EXP $EXP OP_CP |
    OP_OP OP_DIV $EXP $EXP OP_CP |
    OP_OP IDENTIFIER $EXP |
    OP_OP IDENTIFIER $EXP $EXP |
    OP_OP IDENTIFIER $EXP $EXP $EXP|
    IDENTIFIER | VALUEF

7

G++ – Syntax

- Functions:
  - Definition:
    $FUNCTION ->
    (def IDENTIFIER $EXP) |
    (def IDENTIFIER IDENTIFIER $EXP)
    (def IDENTIFIER IDENTIFIER IDENTIFIER $EXP)
  - Parameter passing by value
  - Function definition returns a function value
  - Function application will return the value of the expression evaluated

8

Example Programming in G++

```
$ g++
> (+ 4f1 6f1)
10f1
> (def sum x y (+ x y))
#function
> (sum 4f1 6f1)
10f1
> (exit)
$ _
```

```
$ g++
> (* 4f1 6f1 8f1)
Syntax error!
> (* 4f1 6f1)
24f1
```

9