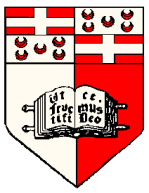
University of Malta



CSA 2010 – Compiling Techniques
Course Assignment 2003-2004

Department of Computer Science and A. I.
University of Malta.
Tutor: Kristian Guillaumier
Email: `kguil@cs.um.edu.mt`

Parsing and Evaluation of BASIC-Style Expression Scripts

- The following is a sample of an expression script:

```
Dim a as integer, b as integer, c as double
c = (10 / a) + b
if a > 10 then
  b = c \ 4
else
  b = c * 4
endif
```

- Design and implement a program that scans, parses, produces a simple symbol table for, and evaluates BASIC-style expression scripts as shown above.
- An expression script may contain variable declaration statements, expression assignment statements, and if-then-else conditional statements.
- Expressions have the following BNF definition:

```
<expression>   →   <primary> {<op> <expression>}
<primary>      →   <identifier>
                 | <integer-constant>
                 | <floatingpoint-constant>
                 | '(' <expression> ')'
<op>           →   '+' | '-' | '*' | '/' | '\'
```

- The operators allowed in the expressions are addition, subtraction, multiplication, floating-point division, and integer division. Unary operators are not permitted.
- Operator precedence is informally defined as follows:

  Multiplication and division are higher than addition and subtraction.

- Brackets may be used to emphasize precedence.
- All operators are left-associative.
- Variable declarations have the following BNF definition:

```
<vardecl>      → 'Dim' <declitem> {',' <declitem>}
<declitem>     → <identifier> 'As' <typename>
<typename>     → 'Integer' | 'Double'
```

- Supported data types in variable declarations are 'Integer' (32-bit signed integer numbers) and 'Double' (32-bit double-precision signed floating-point numbers).
- If-then-else conditional statements have the following BNF definition:

```
<ifstmt>      →   'If' <boolexpr> 'Then'
                  <stmtblock> ['else' <stmtblock>]
                  'Endif'
<boolexpr>    →   <expression> <bop> <expression>
<bop>         →   '>' | '<' | '>=' | '<=' | '<>' | '='
```

- When floating point values are assigned into integer type variables the value is truncated.
- The language is case-insensitive.
- Variable names/identifiers start with a letter or underscore symbol followed by any number of letters, underscore symbols or digits.
- All variables must be declared before being assigned to or used.
- Variables may be assigned to more than once but variable re-declarations are not allowed.
- When the script starts executing, all variables have an undefined value. When a variable is used for the first time (upon reading **not writing** a variable with an undefined value), the user must be asked to enter the value. The value entered must be type-checked. Integers can be implicitly converted to floating-point numbers but not vice versa. An error occurs if a floating-point value is entered for an integer value and the value must be re-submitted.
- The expression script must be read from a text file.
- Once the script file is read, scanned, parsed, and executed, the symbol table must be output to the screen with the final values for all variables. The format should be similar to:

```
Symbol Table
------------

Variable Count: 2

Variable     Type     Value
--------     ----     -----
Counter      INTEGER  6
A            DOUBLE   3.23
```

- The program must produce meaningful error messages including the character and line number where the error occurred.
- The program must be accompanied by a technical report describing any implementation details and techniques used to complete the assignment.
- Refer to the assignment instructions at **http://webster.cs.um.edu.mt/kguil/assignment.html**