A Grammar for the Compiler course project

Fall 2017

1 Introduction

This is the grammar for the fall 2017 semester's Compiler course project. For the grammar that follows here are the types of the various elements by type font or symbol:

- **Keywords are in this type font.**
- ***TOKEN CLASSES ARE IN THIS TYPE FONT.***
- *Nonterminals are in this type font.*
- The symbol ε means the empty string.

1.1 Some Token Definitions

- letter = a | ... | z | A | ... | Z
- digit = 0 | ... | 9
- **SHENASE** = start with a '#', then just two letter characters, and followed by just three digits. e.g. "#zm121", "#ff992". So its length must be exactly 6 characters, neither more nor less than five.

- **ADADSABET** = digit^+
- **REALCONST** = same as a **ADADSABET** but followed by **.ADADSABET**

- **HARF** = is any representation for a single character by placing that character in single quotes. A backslash is an escape character. Any character preceded by a backslash is interpreted as that character. For example \x is the letter x, \' is a single quote, \\ is a single backslash. There are only two exceptions to this rule: \n is a newline character and \0 is the null character.

- **BOOLSABET=** true|false

- **White space** (a sequence of blanks and tabs) is ignored. Whitespace may be required to separate some tokens in order to get the scanner not to collapse them into one token. For example: "intx" is a single **SHENASE** while "int x" is the type **int** followed by the **SHENASE** x. The scanner, by its nature, is a greedy matcher.

- **Comments** are ignored by the scanner. Comments begin with // and run to the end of the line.
- All **keywords** are in lowercase. You need not worry about being case independent since not all lex/flex programs make that easy.

2 The Grammar

1. *program* → **PROGRAM** ***SHENASE*** *declarations_list list_ravie* **MAIN** *block*
2. *declarations_list* → *declarations* | *declarations_list declarations*
3. *declarations* → *taeen_type declarator_list;* | *ε*
4. *taeen_type* → **INT** | **FLOAT** | **CHAR** | **BOOLEAN**
5. *declarator_list* → *declarator* | *declarator_list, declarator*
6. *declarator* → *dec* | *dec := meghdar_avalie*
7. *dec* → ***SHENASE*** | ***SHENASE*** *[range]* | ***SHENASE*** *[**ADADSABET**]*
8. *range* → ***SHENASE*** *..* ***SHENASE*** | ***ADADSABET*** *..* ***ADADSABET*** |
   ebarat_riazi .. ebarat_riazi
9. *meghdar_avalie* → *ebarat_sabet* | { *list_ meghdar_avalie* }
10. *list_meghdar_avalie* → *ebarat_sabet , list_ meghdar_avalie* | *ebarat_sabet*
11. *list_ravie* → *list_ravie ravie* | *ε*
12. *ravie* → **RAVIE** ***SHENASE*** *parameters* { *declarations_list block* };
13. *parameters* → (*declarations_list*)
14. *block* → { *statement_list* }
15. *statement_list* → *statement*; | *statement_list statement*;
16. *statement* → ***SHENASE*** *:= ebarat* |

    AGAR *ebarat_bool* **ANGAH** *statement* |

    AGAR *ebarat_bool* **ANGAH** *statement* **VAGARNA** *statement* |

    DO *statement* **WHILE** *ebarat_bool* |

    FOR ***SHENASE*** *:= counter* **DO** *statement* |

    GOZINESH *ebarat onsor_mored default* **END** |

    SHENASE*(arguments_list)* |

    SHENASE *[ebarat] := ebarat* |

    BAZGASHT *ebarat* |

EXIT WHEN *ebarat_bool* |

block | *ε*

17. *arguments_list* → *multi_arguments* | *ebarat* | *ε*
18. *multi_arguments* → *multi_arguments, ebarat* | *ebarat*
19. counter → ***ADADSABET*** **UPTO** ***ADADSABET*** | ***ADADSABET*** **DOWNTO** ***ADADSABET***
20. *onsor_mored* → **MORED** ***ADADSABET*** : *block* |

    onsor_mored **MORED** ***ADADSABET*** : *block*
21. *default* → *DEFAULT* : *block* | *ε*
22. *ebarat* → *ebarat_sabet* | *ebarat_bool* | *ebarat_riazi* |

    SHENASE | ***SHENASE*** [*ebarat*] | ***SHENASE****(arguments_list)* | (*ebarat*)
23. *ebarat_sabet*→ ***ADADSABET*** | ***REALCONST*** | ***HARF*** | ***BOOLSABET***

    ebarat_bool → zojmoratab **VA** | zojmoratab **YA** | zojmoratab **VA ANGAH**

    | zojmoratab **YA VAGARNA** | zojmoratab< | zojmoratab<=

    | zojmoratab> | zojmoratab>= | zojmoratab = | zojmoratab<>|

    ebarat **NAGHIZ**
24. *ebarat_riazi* → zojmoratab +| zojmoratab - | zojmoratab *| zojmoratab/

    | zojmoratab% | – *ebarat*
25. zojmoratab → *(ebarat, ebarat)*

3 Semantic Notes

- The numbers are **INT**s or **FLOAT**s. Be sure to eliminate leading or trailing zeros.
- There can only be one function with a given name. There is no function overloading.
- In if statements the **VAGARNA** is associated with the most recent **AGAR**.
- Expressions are evaluated in order consistent with operator associativity and precedence found in mathematics.
- Note that **VA** and **YA** evaluate as short-circuit.
- Array assignment works. Array comparison does not. Passing of arrays is done by reference.

- Code that exits a *ravie* without a **BAZGASHT** returns a zero.
- All variables and procedures have to be declared before use.

4 An Example

```
program #ex444
      int #im120 := 1;
      int #jc230 := 2;
      int #kf222 := 3;
      float #rp234 := 1.1;
      boolean #bb212 := true;
      int #ar111[2] := {1, 7};
      char #ch212[i..k] := {'c', 'd', '7'};

      ravie #fu555 (int #in111; boolean #wh121;) {
            int #mi111 := 2;
            int #cd232 := (#in111, 3) +;
            {
                  agar ((#wh121, true)va angah) angah bazgasht
                  (#cd232,#mi111)+;
                  vagarna bazgasht (#cd232, #mi111)–;
            }
      };

      ravie #ab445 (int #in222;) {
            {
                  agar (#in222, 0)> angah bazgasht #in222;
                  vagarna bazgasht (-1, #in222)*;
            }
      }

main {
      #rp234 := (((2, #rp234)*),(3,#rp234)-),4)%;
      do {
            #im120 := #ab445((#im120, #kf222)-);
            exit when ((#ar111[2], 0)>)naghiz
            gozinesh i
                  mored 0: {#im120:= ({#im120, 1)-}
                  mored 1: {#kf222:= ({#kf222, 2)+}
                  mored 2: {#im120:= ({#im120, 5)-}
                  mored 3: {#im120:= ({#im120, 2)+}
                  default: {#kf222:= 0}
            end
            #ar111[2] := (#ar111[2],(#ch212[2] = 'd'))-;
      } while (#kf222,0)>;
      for #jc230:= 10 downto 2 do {
            #im120 := (#im120, 1)+;
            #kf222:= (#kf222, 1)-;
      }
}
```