# 

# The Tiny Language

**Note: (Task (1) deliverable: you will deliver a document containing the RE rules of Tiny Language + DFA + Scanner)**

A program in TINY consists of a set of functions (any number of functions and ends with a main function), each function is a sequence of statements including (declaration, assignment, write, read, if, repeat, function, comment, …) each statement consists of (number, string, identifier, expression, condition, …).

**Language described as:**

1. Number: any sequence of digits and maybe floats (e.g. 123 | 554 | 205 | 0.23 | …)

(digit+) (\.(digit)+)?

1. String: starts with double quotes followed by any combination of characters and digits then ends with double quotes (e.g. “Hello” | “2nd + 3rd” | …)

(\”) ( characters | digits )+ (\”)

1. Reserved\_Keywords: int | float | string | read | write | repeat | until | if | elseif | else | then | return | endl

same

1. Comment\_Statement: starts with /\* followed by any combination of characters and digits then ends with \*/ (e.g. /\*this is a comment\*/ | …)

//(\ /) (\\* ) ( characters | digits)+ (\\* ) (\ /)

1. Identifiers: starts with letter then any combination of letters and digits. (e.g. x | val | counter1 | str1 | s2 | …)

(letter) (letter|digit)\*

1. Function\_Call: starts with Identifier then left bracket “(“ followed by zero or more Identifier separated by “,” and ends with right bracket “)”. (e.g. sum(a,b) | factorial(c) | rand() | … )

//Identifier ( \( ) ((Identifier) ((,)( Identifier))\* )? ( \) )

1. Term: maybe Number or Identifier or function call. (e.g. 441 | var1 | sum(a,b) | …)

//Number | Identifier | function call

1. Arithmatic\_Operator: any arithmetic operation

(\+ | \- | \\* | \ / )

1. Equation: starts with Term or left bracket “(“ followed by one or more Arithmatic\_Operator and Term. with right bracket “)” for each left bracket (e.g. 3+5 | x +1 | (2+3)\*10 | …)

//((Term| \( ) ( \ ) | (Arithmatic\_Operator)(Term))+)+

//((\( )\* Term (Arithmatic\_Operator)(Term))+) ( \ )\*)+

1. Expression: may be a String, Term or Equation (e.g. “hi” | counter | 404 | 2+3 | …)

//String | Term | Equation

1. Assignment\_Statement: starts with Identifier then assignment operator “:=” followed by Expression (e.g. x := 1 //| y:= 2+3 | z := 2+3\*2+(2-3)/1 | …)
2. Datatype: set of reserved keywords (int, float, string)
3. Declaration\_Statement: starts with Datatype then one or more identifiers (assignment statement might exist) separated by coma and ends with semi-colon. (e.g. int x; | float x1,x2:=1,xy:=3; | …)
4. Write\_Statement: starts with reserved keyword “write” followed by an Expression or endl and ends with semi-colon (e.g. write x; | write 5; | write 3+5; | write “Hello World”; | …)
5. Read\_Statement: starts with reserved keyword “read” followed by an Identifier and ends with semi-colon (e.g. read x; | …)
6. Return\_Statement: starts with reserved keyword “return” followed by Expression then ends with semi-colon (e.g. return a+b; | return 5; | return “Hi”; | …)
7. Condition\_Operator: ( less than “<” | greater than “>” | is equal “=” | not equal “<>”)

same

1. Condition: starts with Identifier then Condition\_Operator then Term (e.g. z1 <> 10)
2. Boolean\_Operator: AND operator “&&” and OR operator “||”

same

1. Condition\_Statement: starts with Condition followed by zero or more Boolean\_Operator and Condition (e.g. x < 5 && x > 1)
2. If\_Statement: starts with reserved keyword “if” followed by Condition\_Statement then reserved keyword “then” followed by set of Statements (i.e. any type of statement: write, read, assignment, declaration, …) then Else\_If\_Statment or Else\_Statment or reserved keyword “end”
3. Else\_If\_Statement: same as if statement but starts with reserved keyword “elseif”
4. Else\_Statement: starts with reserved keyword “else” followed by a set of Statements then ends with reserved keyword “end”
5. Repeat\_Statement: starts with reserved keyword “repeat” followed by a set of Statements then reserved keyword “until” followed by Condition\_Statement
6. FunctionName: same as Identifier
7. Parameter: starts with Datatype followed by Identifier   
   (e.g. int x)
8. Function\_Declaration: starts with Datatype followed by FunctionName followed by “(“ then zero or more Parameter separated by “,” then “)” (e.g. int sum(int a, int b) | …)
9. Function\_Body: starts with curly bracket “{” then a set of Statements followed by Return\_Statement and ends with “}”
10. Function\_Statement: starts with Function\_Declaration followed by Function\_Body
11. Main\_Function: starts with Datatype followed by reserved keyword “main” then “()” followed by Function\_Body
12. Program: has zero or more Function\_Statement followed by Main\_Function

**Code Sample**

/\*Sample program includes all 30 rules\*/

int sum(int a, int b)  
{

return a + b;

}

int main()

{

int val, counter;

read val;

counter:=0;

repeat

val := val - 1;

write "Iteration number [";

write counter;

write "] the value of x = ";

write val;

write endl;

counter := counter+1;

until val = 1

write endl;

string s := "number of Iterations = ";

write s;

counter:=counter-1;

write counter;

/\* complicated equation \*/

float z1 := 3\*2\*(2+1)/2-5.3;

z1 := z1 + sum(1,y);  
if z1 > 5 || z1 < counter && z1 = 1 then

write z1;

elseif z1 < 5 then

z1 := 5;

else

z1 := counter;

end

return 0;

}

**Code Sample**

/\* Sample program in Tiny language – computes factorial\*/

int main()

{

int x;

read x; /\*input an integer\*/

if x > 0 then /\*don’t compute if x <= 0 \*/

int fact := 1;

repeat

fact := fact \* x;

x := x – 1;

until x = 0

write fact; /\*output factorial of x\*/

end

return 0;

}