

# Token Definitions

Definitions.

- **Identifier:** Any sequence of characters which may contain alphabets, digits from 0 to 9 or an underscore. The sequence must start with an alphabet or an underscore.
- **Integer:** A sequence of characters which may contain any combination of digits from 0 to 9.
- **White Space:** Any sequence of characters containing any combination of single space, form feed, horizontal tab and vertical tab.
- **Char:** A sequence of three characters which contains a single character between two single quotes. The single character between the two quotes can be any character except a single quote itself. For example, ‘’ is an invalid character.
- **String:** A sequence of characters which contains any number of characters between two double quotes. The character sequence between the two double quotes can be any character except a double quote itself. For example, “Hello”how” is an invalid string.
- **Comment:** Type 1: a sequence of characters which starts with # and ends with a newline. Type2: begins with ‘{’, continues with any character (including end-of-line), and ends with ‘}’.

Pre-defined tokens in the language are given below:

No.	Token	Description
1	\n	Newline
2	program	Start of Program
3	var	Variable
4	const	Constant
5	type	To define a data type
6	function	To define a function
7	return	return from function
8	begin	start of a block
9	end	end of a block
10	::=	swap
11	:=	assignment operator
12	output	output an expression or string
13	if	keyword
14	then	keyword
15	else	keyword
16	while	keyword for loop
17	do	keyword for loop
18	case	keyword
19	of	keyword
20	..	dots for case expression
21	otherwise	keyword
22	repeat	keyword for repeat-until loop

23	<b>for</b>	keyword for loop
24	<b>until</b>	keyword for repeat-until loop
25	<b>loop</b>	keyword for loop-pool loop
26	<b>pool</b>	keyword for loop-pool loop
27	<b>exit</b>	keyword
28	<b>&lt;=</b>	less than equal to binary operator
29	<b>&lt;&gt;</b>	not equal to binary operator
30	<b>&lt;</b>	less than binary operator
		greater than equal to binary
31	<b>&gt;=</b>	operator
32	<b>&gt;</b>	greater than binary operator
33	<b>=</b>	equal to binary operator
34	<b>mod</b>	modulus binary operator
35	<b>and</b>	and binary operator
36	<b>or</b>	or binary operator
37	<b>not</b>	not unary operator
38	<b>read</b>	read an identifier
39	<b>succ</b>	successor of an ordinal value
40	<b>pred</b>	predecessor of a ordinal value
41	<b>chr</b>	keyword for character function
42	<b>ord</b>	keyword for ordinal function
43	<b>eof</b>	keyword for end of file
44	<b>{</b>	begin of a block
45	<b>:</b>	colon
46	<b>;</b>	semi colon
47	<b>.</b>	single dot
48	<b>,</b>	comma
49	<b>(</b>	opening bracket
50	<b>)</b>	closing bracket
51	<b>+</b>	plus
52	<b>-</b>	minus
53	<b>*</b>	multiply
54	<b>/</b>	divide