BSCS-504 COMPILER CONSTRUCTION

SYED MUHAMMAD AAQIB HUSSAIN

TALHA HASAN ZIA

BSCS-III-B

LANGUAGE SPECIFICATIONS

DEPARTMENT OF COMPUTER SCIENCE

**Keywords:**

INTEGER, CHARAC, STRING, IS, RETURN, NULL,

TERMINATE, NEXT, NOT, EQUALS, LESS, GREAT,

ELESS, EGREAT, AND, OR, FUNCTION, CLASS,

CHECK, FROM, IF,THEN,ELSE, UNTIL, RUN,UNTIL, VOID.

\**keywords are always written in all uppercase / capital letters.*

**Data Types:**

**INTEGER:** Signed/Unsigned whole numbers.

**FLOAT:** Real numbers.

**CHARAC:**ASCII valid characters.

**STRING:**String composed of characters.

**Operators:**

**+** Addition.

**-** Subtraction.

**\***  Multiplication.

**/** Division.

**++** Increment.

**--** Decrement.

**Identifiers:**

Identifiers declared in the language are valid if started with a capital alphabet, and can be followed by any number or small alphabet, underscores are allowed but not after first capital letter.

c= Set of capital letters.

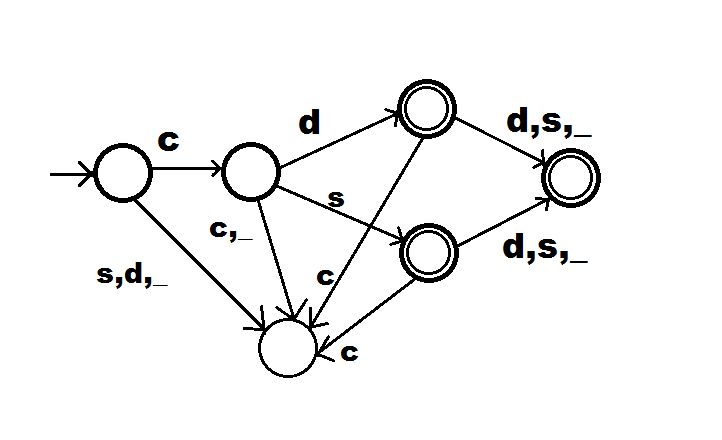
s= Set of small letters.

d= Set of digits 0 to 9.

u= Underscore.

**R.E: (c)(d+s)+(u+d+s)**\*

DFA:



**CONSTANTS:**

**Integer Constants:**

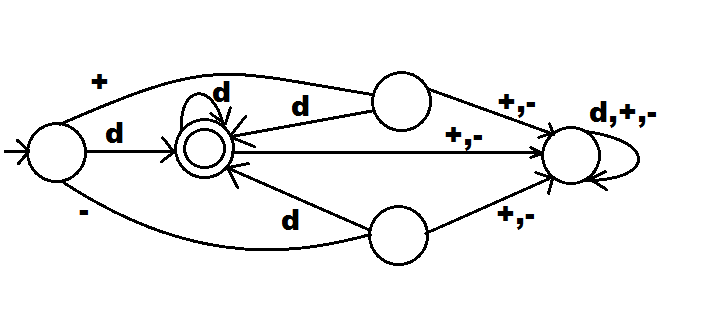
d= Set of digits 0 to 9.

+ = addition sign.

- = subtraction sign.

**R.E: (‘+’ + ‘-‘ + φ)(d)+**

DFA:



**Floating Point Constants:**

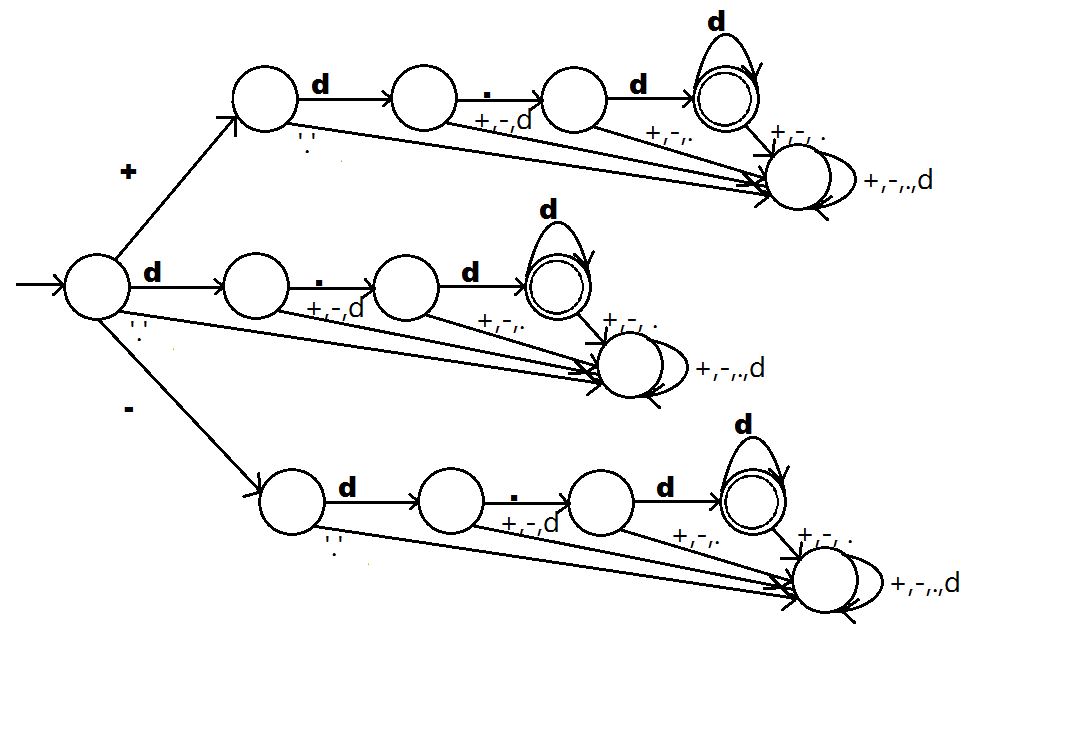
d= Set of digits 0 to 9.

+ = addition sign.

- = subtraction sign.

. = decimal point.

**R.E: (‘+’ + ‘-‘ + φ)(d)+(‘.’)(d)+**

DFA:

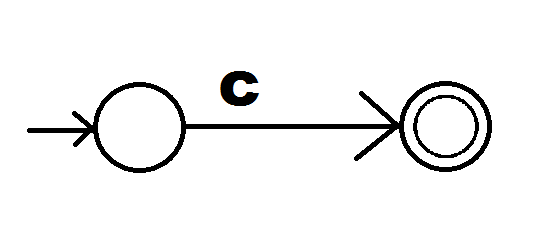
**Character Constants:**

c= Set of all valid ASCII characters.

Note that character data type can hold only one occurance of any ASCII character at a time.

**R.E: (c)**

DFA:



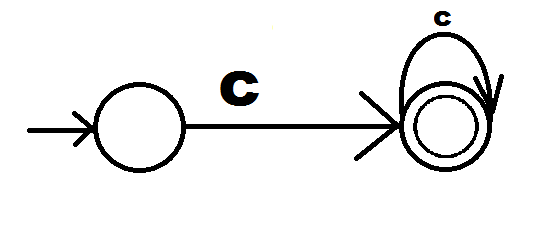
**String Constants:**

c= Set of all valid ASCII characters.

Note that string data type can hold multiple occurance of ASCII characters at a time.

**R.E: (c)+**

DFA:



**SYNTAX:**

**Variable Declarations:**

**All keywords used in uppercase.**

Assignment keyword : **IS**

Decalration:

[Datatype] [Identifier] .

(Note that dot in the end is **terminator**)

Examples:

INTEGER Aint\_1.

FLOAT Fl\_var.

CHARAC Ch\_ch1.

STRING St\_123.

**Variable Initializations:**

Aint\_1 IS 23.

Fl\_var IS -12.47.

Ch\_ch1 IS ‘y’.

St\_123 IS “name of place”.

**OR:**

INTEGER Aint\_1 IS 23.

FLOAT Fl\_var IS -12.47.

CHARAC Ch\_ch1 IS ‘y’.

STRING St\_123 IS “name of place”.

**LOOPS:**

**UNTIL loop:**

Works same as while in C/C++/C#.

Example:

INTEGER Count IS 0.

UNTIL(Count **EQUALS** 10)

{

Count++.

}

*\*runs 10 times.*

**RUN-UNTIL loop:**

Works same as do-while in C/C++/C#.

Example:

INTEGER Count IS 0.

RUN

{

Count++.

}

UNTIL(Count **EQUALS** 10).

*\*runs 11 times.*

**IF-THEN-ELSE Condition:**

Example:

INTEGER Int1 IS 1.

INTEGER Int2 IS 5.

INTEGER Int3 IS 10.

IF(Int1 **EGREAT** 1 **OR** Int3 **ELESS** 10) **THEN**

{

Int1 IS Int2+Int3.

}

**ELSE**

{

Int2++.

}

**CHECK-FROM Condition:**

Works similar to switch.

Example:

CHARAC Ch\_255 IS ‘d’.

**CHECK**(Ch\_255)

{

**FROM** ‘a’:

{

//Implementaion

}

**FROM** ‘b’:

{

//Implementaion

}

**FROM** ‘d’:

{

//Implementaion

}

**ERROR**:

{

//Implementaion

}

}

**FUNCTIONS:**

Functions can be defined anywhere.Function name can only contain alphabets and underscores. Declaration syntax is,

FUNCTION:[functionName](return type, param1, param2, ........ ,paramN)

{

// Implementation

RETURN [variable].

}

*\*if return type set to VOID , above statement is not valid.*

Example:

**FUNCTION:**Add\_Func(INTEGER, INTEGER X1, INTEGER X2)

{

SUM IS X1+X2.

RETURN SUM.

}

**Main Function: (Entry Point)**

FUNCTION:MAIN()

{

// Any logic

}

**Classes: (Optional)**

Syntax,

CLASS [Name]

{

//Implementation

}

Example:

**CLASS** Calculator

{

INTEGER SUM.

FUNCTION:Add\_Func(INTEGER, INTEGER X1, INTEGER X2)

{

SUM IS X1+X2.

RETURN SUM.

}

}

**Other Keywords:**

**TERMINATE =** Works as “Break” in C.

**NEXT =** Works as “Continue” in C.

**NULL =** Works as NULL in C#.

**LESS =** Works as operator < less than.

**GREAT =** Works as operator > greater than.

<IF\_ELSE>**->**IF <RelationalExp> Then OpenBracket <Body> CloseBracket <ElseOp>

<ElseOp>**->**€ | Else OpenBracket <Body> Close