**Tema 2: Automate finite**

Documentatie

# Cerinta

**Partea 1**

Concepeti un program care:

1. Citeste elementele unui automat finit (de la tastatura sau dintr-un fisier, ambele variante trebuie implementate). Descrieti, folosind BNF sau EBNF, cum poate sa arate continutul fisierului (fisier de intrare pentru program) ce contine elementele automatului finit.
2. Afiseaza, prin intermediul unui meniu, urmatoarele elemente ale automatului: multimea starilor, alfabetul, tranzitiile, multimea starilor finale.
3. Pentru un automat finit determinist, verifica daca o secventa este acceptata de automatul finit.
4. Pentru un automat finit determinist, determina cel mai lung prefix dintr-o secventa data care este o secventa acceptata de automat.

Creati un fisier care contine descrierea (conform specificatiei pe care ati stabilit-o individual pt. rezolvarea problemei) unui AFD ce descrie constantelor intregi (literali) din C/C++. Folositi-l ca data de test pentru programul dumneavoastra.

**Partea 2**

1. Rescrieti/adaptati programul de analiza lexicala (tema lab.1) astfel incat sa foloseasca automate finite pentru a determina secventele corespunzatoare atomilor lexicali (folositi automate finite cel putin pentru identificatori, constante numerice). (Programul va obtine FIP, TS)
2. Consultati descrierea constantelor intregi / reale (literali) din limbajul original. Integrati-o in unul dintre cele doua programe de analiza lexicala. NU se vor folosi facilitatile de prelucrare cu expresii regulare (integrate in limbajul in care scrieti aplicatia).

# Observatii referitoare la descrierea constantelor intregi

- consultati documentatia oferita de standardul limbajului in care ati scris primele programe - dati o descriere in unul dintre modurile de specificare neambigue studiate

- in unul dintre programele de analiza lexicala implementate (lab1 sau 2) adaugati identificarea literalilor intregi/virgula\_flotanta conform documentatiei limbajului.

Selectati din descrierea modului de specificare al literalilor toate acele elemente (si numai acele elemente) care sunt prevazute (/implementate) de compilatorul pe care il folositi pentru limbajul din care ati selectat subsetul de instructiuni.

# Specificarea folosind EBNF a fisierului de intrare

input ::= “Q” “:” stari “A” “:” simboluri “T” “:” tranzitii “F” “:” finale “I” “:” initial

stari ::= stare {stare}

stare ::= CONST\_STR

CONST\_STR ::= litera {litera} {cifra}

litera ::= “a”|”b”|”c”| … | “z” |”A”|”B”|…|”Z”

cifra ::= ”0”|“1”|”2”|“3”|”4”|”5”|”6”|”7”|”8”|”9”

simboluri ::= CONST\_CHR {CONST\_CHR}

CONST\_CHR ::= “a”|”b”|”c”| … |”z”|”A”|”B”|…|”Z”|”0”|“1”|”2”|“3”|”4”|”5”|”6”|”7”|”8”|”9”

tranzitii ::= tranzitie {“;” tranzitie}

tranzitie ::= CONST\_STR CONST\_CHR CONST\_STR

finale ::= stare {stare}

initial ::= stare

# Specificarea folosind EBNF a constantelor intregi din C++

CONST\_INT ::= (CONST\_DEC | CONST\_OCT | CONST\_HEX | CONST\_BIN) [sufix]

CONST\_DEC ::= cifra\_nenula {cifra}

cifra\_nenula ::= “1”|”2”|“3”|”4”|”5”|”6”|”7”|”8”|”9”

sufix::= unsigned [long] | long [unsigned]

unsigned ::= “u” | “U”

long ::= “l” [“l”] | “L” [“L”]

CONST\_OCT ::= „0” cifra\_oct {cifra\_oct}

cifra\_oct ::= “0”|“1”|”2”|“3”|”4”|”5”|”6”|”7”

CONST\_HEX ::= ”0” (”X” | ”x”) cifra\_hex {cifra\_hex}

cifra\_hex ::= ”0”|“1”|”2”|“3”|”4”|”5”|”6”|”7”|”8”|”9”|”A”|”B”|”C”|”D”|”E”|”F”| ”a”|”b”|”c”|”d”|”e”|”f”

CONST\_BIN ::= ”0” (”b” | ”B”) cifra\_bin {cifra\_bin}

cifra\_bin ::= ”0”| ”1”

# Implementarea automatului finit

Pentru implementarea automatului finit s-au definit urmatoarele clase:

* **State** – incapsuleaza toate informatiile despre o stare: denumirea, tranzitiile catre alte stari si faptul daca e sau nu finala
* **Alphabet** – abstractizeaza notiunea de „alfabet” al limbajului. Alfabetul contine un vector de litere (string-uri)
* **StateMachine** – incapsuleaza informatii legate de automat: starea initiala, starile automatului, alfabetul, faptul ca e sau nu automat finit. Dandu-se o secventa, automatul poate determina daca aceasta apartine sau nu limbajului. De asemenea, automatul permite extragerea celui mai lung prefix acceptat dintr-o secventa data ca input.

**Verificarea daca o secventa apartine sau nu limbajului**

Inainte de verificarea secventei, se testeaza daca automatul este sau nu finit si daca secventa contine litere din alfabet. In caz negativ, se returneaza false. Daca se trece de aceste validari, urmeaza testarea fiecarei litere din secventa. Se porneste de la starea initiala si se incearca sa se traverseze starile pana la o stare finala, utilizand toate literele din secventa. Daca la un moment dat, dintr-o stare nu se poate trece la alta stare cu simbolul curent, atunci secventa nu este acceptata de automat.

**Determinarea celui mai lung prefix**

Pentru determinarea celui mai lung prefix se testeaza daca secventa apartine automatului. In caz afirmativ, se returneaza intreaga secventa. Altfel, se elimina cel mai din dreapta simbol din secventa si se reia verificarea secventei.

# Automatul finit pentru constante intregi

Diagram

Description automatically generated

# Automatul finit pentru constante string

“

orice

“

# Automatul finit pentru identificatori

a-zA-Z\_

a-zA-Z\_0-9

a-zA-Z\_0-9

a-zA-Z\_0-9

a-zA-Z\_0-9

a-zA-Z\_0-9

a-zA-Z\_0-9

a-zA-Z\_0-9