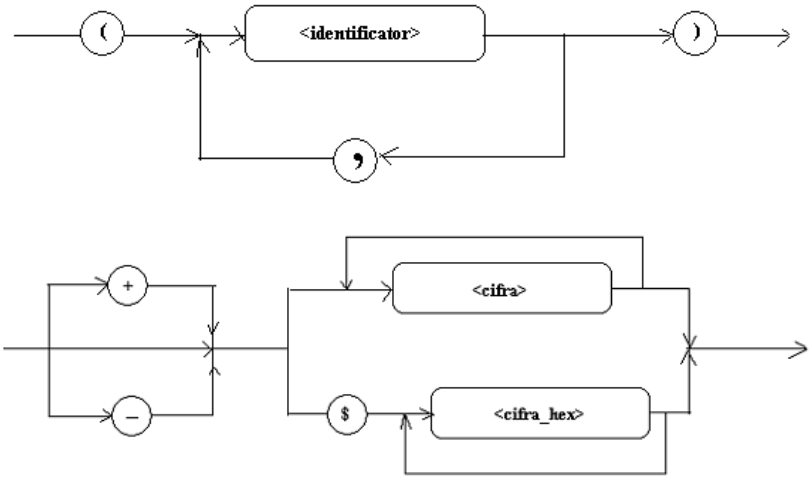
Seminar 1

Minilimbaje de programare. Identificarea si specificarea elementelor lexicale si sintactice.

1. Diagrame de sintaxa

Diagramele de sintaxa sunt reprezentari grafice folosite pentru a descrie sintaxa limbajelor de programare.

1.1. Dati cate 2 exemple valide care respecta urmatoarele specificatii.



1.2. Descrieti, folosind diagrame de sintaxa, sintaxa sectiunii de declaratii de variabile in Pascal. "Neterminalele" (vor aparea in dreptunghiuri cu colturile rotunjite si) vor fi: <ID> si <tip> .

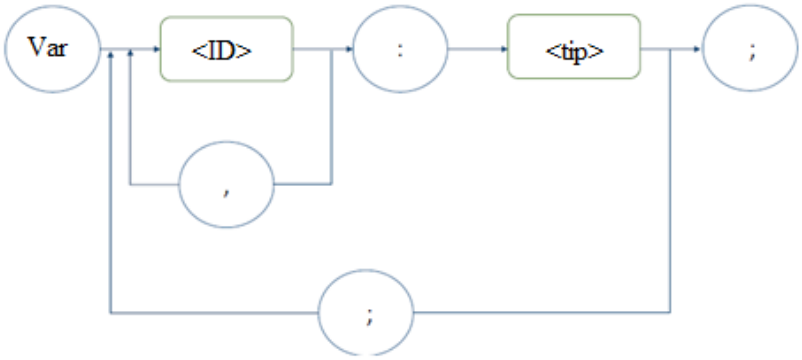
Rezolvari:

Avem diagrama de sintaxa si presupunem ca specificarea elementelor lexicale este cunoscuta.
Astfel, consideram ca:
<identificator> este o succesiune de litere si cifre care incepe cu o litera.
<cifra> este una (oricare) dintre: 0,1,2,3,4,5,6,7,8,9
<cifra_hex> este una (oricare) dintre 0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F

Exemple pentru prima diagrama:
(abc123)
(a,b,c123)
Exemple pentru a doua diagrama:
+1234
-$ABCDEF
7

1.2
Exemplu:
```
Var     a,b,c: integer;
        x: real;
```
Sectiunea incepe cu cuvantul var , dupa care urmeaza listele de declaratii. O lista de declaratii consta dintr-o lista de variabile separate prin virgula, urmate de ":" apoi de tipul lor si se incheie cu ";"

In locul numelor variabilelor vom folosi <ID>, iar in locul tipului lor vom folosi: <tip>



2. BNF si EBNF

2.1. Dati o descriere echivalenta in BNF si EBNF corespunzatoare diagramelor din sectiunea precedenta.

3. Elemente lexicale si sintactice ale unui limbaj de programare

3.1. Fie o gramatica ce descrie sintaxa unui mini-limbaj de programare, data prin regulile de productie:

<program> → begin <lista_instr> end .
<lista_instr> → <instr> ; <lista_instr>
<lista_instr> → <instr>
<instr> → <atribuire>
<instr> → <instr_if>
<atribuire> → ID = <expr>
<expr> → <expr> + <variabila>
<expr> → <variabila>
<variabila> → ID
<instr_if> → if (<expr>) then <atribuire>

---

a) Identificati terminalele si neterminalele gramaticii
b) Dati doua "mini-programe" care sunt descrise de specificatiile date.
ID este un atom lexical. Regulile de formare a acestui atom lexical ID (identificator), specificate folosind expresii regulare, sunt:
a (a | b | c) *

Rezolvare:

a)

Terminale:
"begin" , "end", "." ; "ID", "=" , "+" , "if" , "then", "(" , ")"
Neterminale:
<program> , <lista_instr> , <instr> , <atribuire> , <expr> , <variabila> , <instr_if>

b)

begin

```
abc = cc;
if (cc) then abcabc=abc
end.

begin
aa=ab;
ac=ab+ac
end.
```

3.2. Fie urmatorul exemplu de program Pascal:

```
var  f, a1, a2, a3 : integer;
begin
        a1:= 7;
        a2:= 11;
        a3:= a1+a2+3;
        f  := 5
end.
```

a) Identificati elementele lexicale si structurile sintactice.
b) Descrieti sintaxa structurilor sintactice folosind unul dintre mecanismele de specificare: BNF sau EBNF.
c) Scrieti un program diferit de cel de mai sus care respecta descrierile date.

d) Presupunand ca operatorii si cuvintele cheie din exemplul de mai sus au asociate coduri – numere naturale in ordine crescatoare,
in ordinea in care ele apar in program,
descrieti continutul tabelei FIP, precum si a tabelei de simboluri, atunci cand se folosesc 2 tabele de simboluri, una pentru constante, una pentru identificatori, pentru fiecare dintre urmatoarele 3 organizari:
   i. Tabel sortat lexicographic
   ii. Arbore binar de cautare
   iii. Tabela de dispersie
   (Alegeti o functie de dipersie simplu de calculat, alegeti dimensiunea tabelei ca fiind 11.)

Poate aveti nevoie:
ASC('a') = 97
ASC('0') = 48

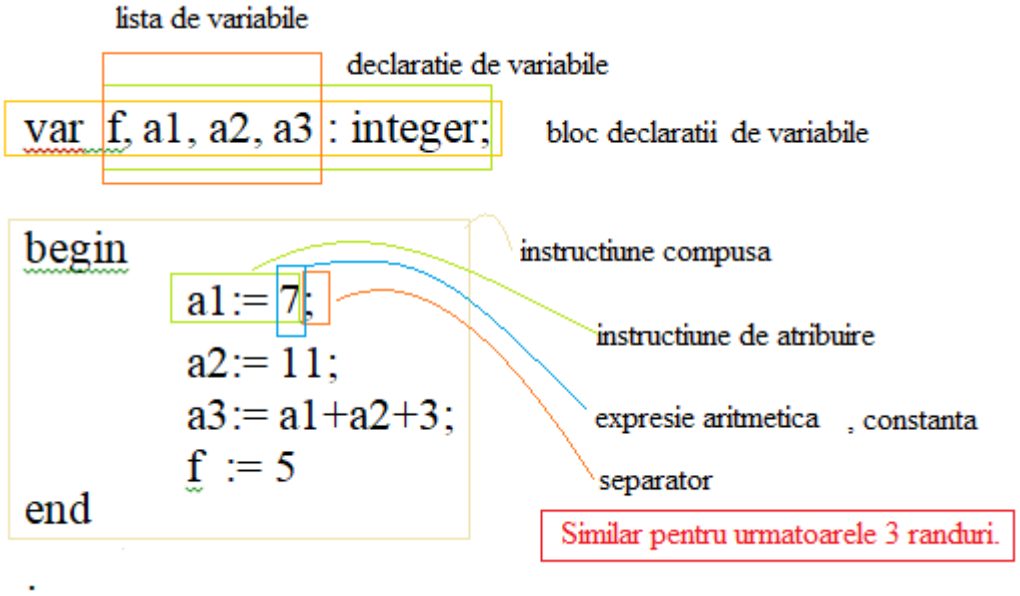
Rezolvare:

Elemente lexicale:

Cuvinte rezervate/cheie:	var, integer, begin, end
ID:	a1, a2, a3, f
Operatori:	":=", "+"
Delimitatori/separatori:	";" , ":" , "." , " ", ",", "\n"
CONST:	7, 11, 5, 3

Obs: In pascal, integer nu este cuvant cheie, dar, pentru simplificare, noi il vom considera cuvant cheie. Se poate folosi aceasta simplificare si in cadrul temelor de laborator.

Structuri sintactice:



.

b) specificare in BNF

<program> ::= <bloc-decl-var> <instr-comp> .
<bloc-decl-var> ::= var <lista-decl-var>
<lista-decl-var> ::= <decl-var> | <decl-var> <lista-decl-var>
<decl-var> ::= <lista_variabila> : integer;
<lista_variabila> ::= ID , <lista_variabila> | ID
<instr-comp> ::= begin <lista_instr> end
<lista_instr> ::= <instr> ; <lista_instr> | <instr>
<instr> ::= <atribuire>
<atribuire> ::= ID := <expr>
<expr> ::= <expr> + <expr> | CONST | ID

c)

```
var     a1 : integer;
        a2, a3 : integer;
begin
        a1:= a2+a3 + 7
end.
```

d) i.

Observatie: ID si CONST sunt atomi lexicali speciali, pentru ei vom avea codurile 0 si 1.

Atom lexical	Cod Atom
ID	0
CONST	1
var	2

,	3
:	4
integer	5
;	6
begin	7
:=	8
+	9
end	10
.	11

Obs.: In FIP am colorat cu galben pozitia corespunzatoare “;” doar cu scopul de a ne urmari mai usor pozitia curenta din program.

FIP	
Cod Atom	***Pozitie TS***
2	
0	4
3	
0	1
3	
0	2
3	
0	3
4	
5	
6	
7	
0	1
8	
1	3
6	
0	2
8	
1	4
6	
0	3
8	
0	1
9	
0	2
9	
1	1
6	
0	4
8	
1	2
10	
11	

TS pentru ID

Pozitie in tabel	Simbol (ID)	Alte atribute
1 2 3 4	a1 a2 a3 f	Se memoreaza si alte atribute (daca e cazul)

Pozitia in tabel nu se memoreaza. Poate incepe de la 0 sau de la 1.

TS pentru CONST

Poz. in tabel	Simbol (CONST)	Alte atribute
1 2 3 4	3 5 7 11	Se memoreaza si alte atribute (daca e cazul)

Observatii:
Tabelele TS pentru identificatori si constante sunt ordonate lexicografic.

O a doua varianta de constructie a acestor tabele este de a folosi campuri de legatura/inlantuire. Sa luam exemplul tabelei identificatorilor. Astfel, la construirea tabelei, noii identificatori vor fi adaugati la sfarsit, dar vom avea o coloana suplimentara in tabel care va indica urmatorul element in ordine lexicografica. In acest caz, tabelul va arata astfel (iar valorile din FIP folosite pentru identificarea atomilor vor fi altele):

TS pentru ID

Pozitie in tabel	Simbol (ID)	Legatura ordine lexicografica
1	f	-1
2	a1	3
3	a2	4
4	a3	1

Astfel, avem o lista simplu inlantuita ordonata, reprezentata pe tabel, iar pozitia 2 este pozitia capului listei.

In cazul in care optam pentru a reprezenta inlantuit pe tabel arborele binar de cautare, tabelul va arata astfel:

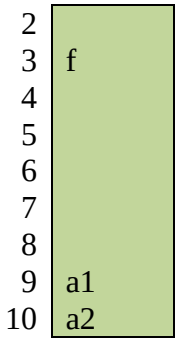
TS pentru ID

Pozitie in tabel	Simbol (ID)	Legatura stanga	Legatura dreapta
1	f	2	-1
2	a1	-1	3
3	a2	-1	4
4	a3	-1	-1

iii) Pentru reprezentare folosind tabela de dispersie, trebuie sa alegem o functie de dispersie si o strategie de rezolvare a coliziunilor.

Fie:
m=dimensiunea tabelei de dispersie
funcția de dispersie = (codul ascii al primului character) mod m
cu: open addressing & linear probing
Daca m=11, tabela de simboluri pentru identificatori va fi:

Pozitie in tabel	Simbol (ID)
0	a3
1	


f
a1
a2