**LFTC – MLP**

Laborator1

1.

- Tipuri de Date:

* int: Tip de date utilizat pentru a reprezenta numere întregi, curpinse între -2.147.483.648 și 2.147.483.647 (pe 32 de biți)
* float: Tip de date folosit pentru a reprezenta numere reale (cu virgulă mobilă) cu precizie simplă, adică pentru valori numerice care pot avea zecimale
* struct: Tip de date definit de utilizator, care grupează mai multe variabile de tipuri diferite sub un sigur nume

- Instrucțiuni:

* **Instrucțiune de atribuire**: variabilă = expresie;
* **Instrucțiune de intrare/ieșire**: cin >> variabilă; cout << expresie;
* **Instrucțiune de selecție**:

if (condiție) {

instrucțiuni\_true

}

else {

instrucțiuni\_false

}

* **Instrucțiune de ciclare**:

for (inițializare; condiție; incrementare) {

instrucțiuni

}

- Restricții:

* **Identificatori**: Vor începe obligatoriu cu o literă (a-z sau A-Z) și pot conține litere, cifre (0-9) și caracterul de subliniere (\_). Nu pot începe cu o cifră. Lungimea unui identificator este limitată la 20 de caractere pentru a facilita gestionarea variabilelor în analizatorul lexical. Cuvintele rezervate din C++ nu pot fi folosite ca identificatori. Sunt case-sensitive.
* **Constante**: Constantele trebuie să aibă maximum 30 de caractere și să nu conțină caractere speciale, cu excepția spațiilor.
* **Comentarii**: Se permite doar comentarii pe o singură linie folosind //. Comentariile multi-linie nu sunt permise.
* **Declarații**: Se vor face doar la inceputul programului și o variabilă pe linie. Struct-urile se declară înainte de funcția main.

**Analizator Lexical**:

Simboluri utilizate:

1. Operatori:

- Aritmetici: “+”, “-”, “\*”, “/”, “%”

- Relationali: “<”, “<=”, “>”, “>=”, “==”, “!=”

- Atribuire: “=”

2. Cuvinte cheie:

- Tipuri de date permise: “int”, “float”, “struct”

- Structuri de control: “if”, “else”, “for”

- Instructiuni de intrare/iesire: “cin >>”, “cout <<”

- Start program: “main”

- Linie nouă: “endl”

3. Identificatori:

- identificator = litera { litera | cifra | “\_” }

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

- cifra = “0” | “1” | … | “9”

4. Delimitatori: “;”, “,”, “(“, “)”, “{“, “}”

EBNF:

program = declarații\_struct "void main() {" {declarații\_var} {instrucțiuni} "}".

declarații\_struct = “struct” ID “{“ declaratii\_var “}”.

ID = literă {literă | cifră | “\_”}.

declaratii\_var = tip\_de\_date ( ID | instr\_atribuire) “;”.

tip\_de\_date = ”int” | ”float” | ”struct”.

instrucțiuni = (instr\_intrare | instr\_ieșire | instr\_atribuire | instr\_control | instr\_ciclare) .

instr\_atribuire = ID ”=” expresie ”;”.

instr\_intrare = ”cin” ”>>” ID {”>>” ID} ”;”.

instr\_ieșire = “cout” “<<” (expresie | “endl”) {“<<” (expresie | “endl”)} ”;”.

instr\_control = “if(” condiție “){” {instrucțiuni} ”}” [”else{” {instrucțiuni} ”}”].

instr\_ciclare = ”for(” [instr\_atribuire] ”;” condiție “;” [instr\_atribuire] “){“ {instrucțiuni} “}”.

expresie = CONST | ID | operație.

operație = (ID | CONST) operator\_artimetic ((ID | CONST) | {operație}).

condiție = expresie operator\_relațional expresie.

CONST = nr\_întreg | nr\_real | “”” {literă} “””.

nr\_întreg = [“-“] cifră\_nenulă {cifră}.

nr\_real = nr\_întreg [“.” {cifră}].

literă = “a” | “b” | … | “z” | “A” | “B” | … | “Z”.

cifră = “0” | “1” | … | “9”.

cifră\_nenulă = “1” | … | “9”.

operator\_aritmetic = “+” | “-” | “\*” | “/” | “%”.

operator\_relațional = “<” | “<=” | “>” | “>=” | “==” | “!=”.

|  |  |
| --- | --- |
| **Atom** | CodAtom |
| ID | 0 |
| CONST | 1 |
| void | 2 |
| main | 3 |
| ( | 4 |
| ) | 5 |
| { | 6 |
| } | 7 |
| ; | 8 |
| , | 9 |
| struct | 10 |
| int | 11 |
| float | 12 |
| = | 13 |
| cin | 14 |
| >> | 15 |
| cout | 16 |
| << | 17 |
| endl | 18 |
| if | 19 |
| else | 20 |
| for | 21 |
| + | 22 |
| - | 23 |
| \* | 24 |
| / | 25 |
| % | 26 |
| < | 27 |
| <= | 28 |
| > | 29 |
| >= | 30 |
| == | 31 |
| != | 32 |