**MLP: C++**

<program> ::= <lista\_importuri> using namespace std; int main() {<lista\_instructiuni> return 0;}

<lista\_importuri> ::= <import> | <import><lista\_importuri>

<import> ::= #include < <librarie> >

<librarie> ::= iostream | math.h

**Tipuri de date**

**<sir\_caractere>** ::= <ch><sir\_caractere> | <ch>

<ch> ::= a|b|c|…|z

//zecimal

**<numar\_intreg>**  ::= <numar\_natural> | <nenul>

<numar\_natural> ::= 0 | <nenul >

<nenul> ::= <cifra\_nenula> | <cifra\_nenula> <sir\_cifre>

<sir\_cifre> ::= <cifra> |< cifra> <sir\_cifre>

<cifra> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<cifra\_nenula> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

**<numar\_intreg\_binar>** ::= 0b<cifra\_binara> | 0b<cifra\_binara><sir\_cifre\_binare>

<sir\_cifre\_binare> ::= <cifra\_binara> | <cifra\_binara><sir\_cifre\_binare>

<cifra\_binara> ::= 0 | 1

**<numar\_intreg\_octal>** ::= 0<cifra\_octal> | 0<cifra\_ octal ><sir\_cifre\_ octal >

<sir\_cifre\_ octal > ::= <cifra\_ octal > | <cifra\_ octal ><sir\_cifre\_ octal >

<cifra\_ octal > ::= 0 | 1 | .. | 7

**<numar\_intreg\_hexa>** ::= 0x<cifra\_hexa> | 0x<cifra\_ hexa ><sir\_cifre\_ hexa >

<sir\_cifre\_ hexa > ::= <cifra\_ hexa > | <cifra\_ hexa ><sir\_cifre\_ hexa >

<cifra\_ hexa > ::= 0 | 1 | .. | 9 | A | B | C | D | E | F

**<numar\_rational>** ::= <numar\_intreg > | < numar\_intreg >.<numar\_natural>

**<Persoana>** ::= <Nume> <CNP>

<Nume> ::= <sir\_caractere> <Nume> | <sir\_caractere>

<CNP> ::= <cifra><cifra><cifra><cifra><cifra><cifra><cifra><cifra><cifra><cifra>

* 1. **Instructiuni**

<lista\_instructiuni> ::= <instructiune><lista\_instructiuni> | <instructiune>

<instructiune> ::= <atribuire> | <ciclare> | <intrare> | <iesire> | <conditional> | <declarare>

**<atribuire>** ::= <id> = <expr>;

<expr> ::= <id> | <const> | <expresie> <op> <expresie>

<op> ::= + | - | \* | / | %

**<ciclare>** ::= <while> | <for>

<while> ::= while (<conditie>) { <lista\_instructiuni> }

<conditie> ::= <expr> <relatie> <expr>

<relatie> ::= == | != | < | <= | > | >=

<for> ::= for (<atribuire>; <conditie>; <incrementare>) { <lista\_instructiuni> }

<incrementare> ::= <id>++

**<intrare>** ::= cin>> <id>;

**<iesire>** ::= cout<< <expr>;

**<conditional>** ::= if (<conditie>) { <lista\_instructiuni> } <altfel>

<altfel> ::= else { <lista\_instructiuni> } | ε

**<declarare>** ::= <tip> <id>; | <tip> <id> = <const>;

<tip> ::= int | float | char

<id> ::= <sir\_caractere> //de completat

**<const>** ::= <numar\_intreg> | <numar\_intreg\_binar> | <numar\_intreg\_octal> | <numar\_intreg\_hexa>| <numar\_rational> | “sir\_caractere”

1. **Programe**
   1. **arie și perimetru**

#include <iostream>

using namespace std;

int main(){

float pi;

float p;

float a;

float r;

pi = 3.14159;

cin>>r;

p=2\*pi\*r;

a=pi\*r\*r;

cout<<p;

cout<<a;

return 0;

}

* 1. **cmmdc**

#include <iostream>

using namespace std;

int main() {

int a, b;

cin >> a;

cin >> b;

while (b != 0) {

int aux = b;

b = a % b;

a = aux;

}

cout << a;

return 0;

}

* 1. **suma a n nr citite de la tastatura**

#include <iostream>

using namespace std;

int main() {

int n;

int suma = 0;

cin >> n;

if (n <= 0) {

cout << "Numar invalid de elemente.";

}

for (int i = 0; i < n; i++) {

int numar;

cin >> numar;

suma = suma + numar;

}

cout << suma;

return 0;

}

**3. programe cu erori**

**3.1 conform mlp-ului, dar si limbajului original**

#include <iostream>

using namespace std

int main() {

int x = 10 //lipseste ;

int y = 0;

int rezultat = x / y

cout <<<rezultat;

return 0;

}

**3.2 nefiind erori in limbajul original**

#include <iostream>

using namespace std;

int main() {

int a, b;

cin >> a>>b; //nu am definit asta

cout << a+b;

return 0;

}