Documentatie lab 6 LFTC

- Detalierea cerintelor pentru “limbajul minimal”
  - Extended Backus-Naur Form
- Restricții
- Textele sursă pentru cele 2 mini-programe corecte
  - Program #1
  - Program #2

Detalierea cerintelor pentru “limbajul minimal”

Extended Backus-Naur Form

program = program_header “{“ lista_decl “return” CONST ”;” “}“.
program_header = “#include” “<iostream>” tip “main” “(“ “)”.
lista_decl = decl [“;”] { decl [“;“] }.
decl = variabila_decl | attr_decl | cin_instr | cout_instr.
variabila_decl = tip ID { “,” tip ID } “;”.
tip = “int”.
attr_decl = ID “=“ expr “;”.
cin_instr = “std::cin” { “>>“ ID } “;”.
cout_instr = “std::cout” { “<<“ (ID | “std::endl“) } “;”.
expr = expr_term { operator expr_term }.
expr_term = expr | ID | CONST.
operator = "+" | "-" | "*" | "/".

Restricții

ID - să înceapă cu o literă și să fie format doar din litere și cifre.
CONST - poate fi orice caracter fără apostrof simplu și dublu.

Textele sursă pentru cele 2 mini-programe corecte

Program #1

```
#include <iostream>

int main()
{
    int a, b, c;

    std::cin >> a;
    std::cin >> b;

    c = a + b * 3;

    std::cout << std::endl;
    std::cout << c;

    return 0;
}
```

Program #2

```cpp
#include <iostream>

int main()
{
    int a, b, c, d;

    std::cin >> a;
    std::cin >> b;

    a = a * 3 - 3;
    b = 1 + a - 2 * b / 4;
    c = 5;
    d = a * 3 + b * 2 + c;

    std::cout << std::endl;
    std::cout << d;

    return 0;
}
```