Завдання 1

Описати мову, яка описується регулярним виразом



#include <iostream> #include <regex>

bool is\_valid\_string(const std::string& str) { std::regex re("((ε|a)·(c,b))+");

return std::regex\_match(str, re);

}

int main() { std::string input;

std::cout << "Введіть рядок: "; std::cin >> input;

if (is\_valid\_string(input)) {

std::cout << "Рядок належить мові.\n";

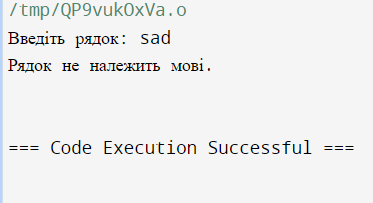
} else {

std::cout << "Рядок не належить мові.\n";

}

return 0;

}



Завдання 2

Написати регулярні визначення для наступних мов:



#include <iostream> #include <regex>

bool is\_valid\_string(const std::string& str) { std::regex re("^[0-9]+$");

return std::regex\_match(str, re);

}

int main() { std::string input;

std::cout << "Введіть рядок: "; std::cin >> input;

if (is\_valid\_string(input)) {

std::cout << "Рядок належить мові.\n";

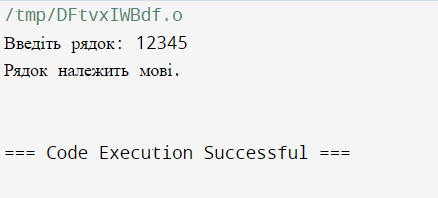
} else {

std::cout << "Рядок не належить мові.\n";

}

return 0;

}



Завдання 3

Використовуючи стандарт мови написати вхідну абетку для мови програмування, лексем ключових слів (до 20), лексеми односимвольних і багатосимвольні роздільників, лексеми арифметичних операцій, лексеми логічних і бітових операцій):



import re

KEYWORDS = [

"and", "as", "assert", "break", "class", "continue", "def", "del",

"elif", "else", "except", "finally", "for", "from", "global", "if",

"import", "in", "is", "lambda", "not", "or", "pass", "print", "raise", "return", "try", "while", "with", "yield",

]

SINGLE\_DELIMITERS = ["(", ")", "[", "]", "{", "}", ",", ";", ".", ":", "?", "!", "=",

"+", "-", "\*", "/", "%", "&", "|", "^", "~", "<", ">", "<=", ">=", "==", "!=", "<>"]

MULTI\_DELIMITERS = [("\*/", re.compile(r"/\\*.\*"))]

ARITHMETIC\_OPERATORS = ["+", "-", "\*", "/", "//", "%", "\*\*"]

LOGICAL\_OPERATORS = ["and", "or", "not"]

BITWISE\_OPERATORS = ["&", "|", "^", "~", "<<", ">>"]

def tokenize(source\_code): tokens = []

for match in re.finditer(r"[^\s]+", source\_code):

token = match.group()

if token in KEYWORDS: tokens.append((token, "keyword"))

elif token in SINGLE\_DELIMITERS: tokens.append((token, "delimiter"))

elif token in MULTI\_DELIMITERS:

for delimiter, regex in MULTI\_DELIMITERS: if regex.match(token):

tokens.append((delimiter, "delimiter")) break

elif token in ARITHMETIC\_OPERATORS: tokens.append((token, "arithmetic\_operator"))

elif token in LOGICAL\_OPERATORS: tokens.append((token, "logical\_operator"))

elif token in BITWISE\_OPERATORS: tokens.append((token, "bitwise\_operator"))

else:

tokens.append((token, "identifier")) return tokens

#Приклад використання словника programmer\_input = "x = 2 + 3 \* y" tokens = tokenize(programmer\_input) print(tokens)

