Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине: «Языковые процессоры интеллектуальных систем»

Тема: «Проектирование лексического анализатора**»**

Выполнил:

Студент 3 курса

Группы ИИ-16

Пастушик М.С.

Проверил:

Слинко Е.В.

Брест 2021

**Цель работы:** изучение основных понятий теории регулярных грамматик, ознакомление с назначением и принципами работы лексических анализаторов (сканеров), получение практических навыков построения сканера на примере заданного простейшего входного языка.

**Вариант 5:**

Входной язык содержит выражения над строковыми константами, разделенные символом ; (точка с запятой). Выражения состоят из идентификаторов, строковых констант, заключенных в двойные кавычки, одиночных символов, заключенных в одинарные кавычки и знаков операции конкатенации +.

**Текст программы:**

#include <iostream>

#include <iomanip>

#include <fstream>

#include <regex>

#include <vector>

void printSpellingError(const std::string);

std::vector<std::string> getLexems(const std::string , std::regex);

int main() {

std::vector<std::string> lines;

std::string str;

std::ifstream file("lab2/file.txt");

while(std::getline(file, str)) {

lines.push\_back(str);

}

for(auto& text : lines) {

text = std::regex\_replace(text, std::regex(" "), "");

std::smatch res;

std::regex re("(['\"])(.\*)(['\"])(\\+)(['\"])(.\*)(['\"])(;)");

if(std::regex\_match(text, res, re)){

for(size\_t i = 1; i < res.size(); i++){

if(res[i].str() == "'") std::cout << res[i].str() << std::string(10, ' ') << "Single quotes" << '\n';

else if(res[i].str() == "\"") std::cout << res[i].str() << std::string(10, ' ') << "Double quotes" << '\n';

else if(res[i].str() == ";") std::cout << res[i].str() << std::string(10, ' ') << "Semicolon" << '\n';

else if(res[i].str() == "+") std::cout << res[i].str() << std::string(10, ' ') << "Concatenation" << '\n';

else if(res[i].str().length() > 1) std::cout << res[i].str() << std::string(10, ' ') << "String" << '\n';

else if(res[i].str().length() == 1) std::cout << res[i].str() << std::string(10, ' ') << "Symbol" << '\n';

}

}

else {

std::cout << "Spelling error\n";

printSpellingError(text);

}

}

return 0;

}

void printSpellingError(const std::string text){

std::vector<std::string> dlms = getLexems(text, std::regex("[a-z]|[A-Z]|[0-9]"));

std::vector<std::string> str = getLexems(text, std::regex("[\"]|[+]|[;]|[']"));

bool isString = false;

bool isSymbol = false;

bool isSemicolon = true;

bool shouldBeStrFirst = true;

bool shouldBeStrNext = true;

bool shouldBePlus = false;

for (size\_t i = 0, j = 0; i < str.size() || j < dlms.size(); i++, j++) {

if (j < dlms.size())

for (const auto& delimeter : dlms[j]) {

if (delimeter == '\"') {

shouldBeStrFirst = false;

shouldBePlus = true;

if(isString) isString = false;

else isString = true;

if(shouldBeStrNext) shouldBeStrNext = false;

}

else if (delimeter == '\'') {

shouldBePlus = true;

shouldBeStrFirst = false;

if(isSymbol) isSymbol = false;

else isSymbol = true;

if(shouldBeStrNext) shouldBeStrNext = false;

}

else if (delimeter == '+') {

shouldBeStrNext = true;

shouldBePlus = false;

if(shouldBeStrFirst){

std::cout << "[ERROR]String or symbol must be placed before +\n";

return;

}

if(!isSemicolon) {

std::cout << "[ERROR]Semicolon expected\n";

return;

}

else isSemicolon = false;

}

else if (delimeter == ';') {

if(shouldBeStrFirst){

std::cout << "[ERROR]String or symbol must be placed before ;\n";

return;

}

else if(shouldBePlus){

std::cout << "[ERROR]+ must be placed after string\n";

return;

}

else if(shouldBeStrNext){

std::cout << "[ERROR]String or symbol must be placed after +\n";

return;

}

isSemicolon = true;

}

else {

std::cout << "[ERROR] Unexpected symbol\n";

return;

}

}

if (i < str.size()) {

if(isSymbol && str[i].size() != 1){

std::cout << "[ERROR]String should be placed in double qoutes\n";

return;

}

}

}

}

std::vector<std::string> getLexems(const std::string s, std::regex rgx) {

std::vector<std::string> elems;

std::sregex\_token\_iterator iter(s.begin(), s.end(), rgx, -1);

std::sregex\_token\_iterator end;

while (iter != end) {

if(\*iter != "") elems.push\_back(\*iter);

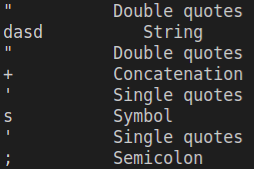
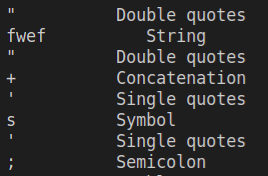
++iter;

}

return elems;

}

**Результат:**



**Пример ошибок:**





**Вывод:** в ходе лабораторной работы были изучены основные понятия теории регулярных грамматик, назначение и принципы работы лексических анализаторов (сканеров), получены практические навыки построения сканера на примере заданного простейшего входного языка.