УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №1

по предмету

Компиляторные технологии

Вариант 5

Выполнили:

Наривончик А. М.

Группа 351004

Проверила:

Болтак С. В.

Минск 2024

**Задание:**

Двоичное, восьмеричное или 16-чное число в синтаксисе FASM (постфиксная форма записи). 16-чное число должно начинаться с цифры от 0 до 9.

**Примеры корректных строк:**

01101011B

54212110o

51245H

0B8Ch

**Регулярное выражение**

([01]+ [Bb])|(([01]|[2-7])+ [Oo])|([01]|[2-7]|[89])([01]|[2-7]|[89]| [AaC-Fc-f]|[Bb])\* [Hh])

**Константы:**

Binary ::= [01]

Octal ::= [2-7]

Decimal ::= [89]

Hexadecimal ::= [AaC-Fc-f]

BinaryMark ::= [Bb]

OctalMark ::= [Oo]

HexMark ::= [Hh]

BinaryLiteral ::= Binary+ BinaryMark

OctalLiteral ::= (Binary | Octal)+ OctalMark

HexLiteral ::= (Binary | Octal | Decimal) (Binary | Octal | Decimal | Hexadecimal | BinaryMark)\* HexMark

Literal ::= BinaryLiteral | OctalLiteral | HexLiteral

**НКА (для регулярного выражения)**



**ДКА (для регулярного выражения)**



**ДКА(готовый к программной реализации)**



**Таблица переходов**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Binary** | **Octal** | **Decimal** | **Hexadecimal** | **BinaryMark** | **OctalMark** | **HexMark** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **1** | **2** | **3** | **4** | **0** | **0** | **0** | **0** |
| **2** | **2** | **3** | **4** | **7** | **5** | **6** | **8** |
| **3** | **3** | **3** | **4** | **7** | **7** | **6** | **8** |
| **4** | **4** | **4** | **4** | **7** | **7** | **0** | **8** |
| **5** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **6** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **7** | **7** | **7** | **7** | **7** | **7** | **0** | **8** |
| **8** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

|  |  |
| --- | --- |
| **Состояние** | **Значение** |
| 0 | Состояние «ошибка» |
| 1 | Прочитано: ничего  Ожидается: десятичная цифра |
| 2 | Прочитано: цифра 0 или 1  Ожидается: 16-ричная цифра или признак окончания двоичного литерала |
| 3 | Прочитано: цифра от 2 до 7  Ожидается: 16-ричная цифра или признак окончания восьмеричного литерала |
| 4 | Прочитано: цифра 8 или 9  Ожидается: 16-ричная цифра или признак окончания 16-ричного литерала |
| 5 | Прочитано: признак конца двоичного литерала b или B  Ожидается: ничего |
| 6 | Прочитано: признак конца восьмеричного литерала o или O  Ожидается: ничего |
| 7 | Прочитано: 16ричная цифра  Ожидается: 16ричная цифра или признак окончания 16-ричного литерала |
| 8 | Прочитано: признак окончания 16-ричного литерала h или H  Ожидается: ничего |

**Код программы, реализующий работы ДКА**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

enum tCharType {ctUnknown, ctBinary, ctOctal, ctDecimal, ctHexadecimal, ctBinaryMark, ctOctalMark,   
 ctHexMark};

typedef int tState;

const tState transitions[9][8] = {

{0, 0, 0, 0, 0, 0, 0, 0}, //0 ошибка

{0, 2, 3, 4, 0, 0, 0, 0}, //1

{0, 2, 3, 4, 7, 5, 6, 8}, //2

{0, 3, 3, 4, 7, 7, 6, 8}, //3

{0, 4, 4, 4, 7, 7, 0, 8}, //4

{0, 0, 0, 7, 7, 7, 0, 8}, //5

{0, 0, 0, 0, 0, 0, 0, 0}, //6

{0, 7, 7, 7, 7, 7, 0, 8}, //7

{0, 0, 0, 0, 0, 0, 0, 0}, //8

};

const bool isFinalState[9] = {false, false, false, false, false, true, true, false, true};

tCharType getCharType(char charToCheck) {

if (charToCheck >= '0' && charToCheck <= '1') return ctBinary;

if (charToCheck >= '2' && charToCheck <= '7') return ctOctal;

if (charToCheck >= '8' && charToCheck <= '9') return ctDecimal;

if (charToCheck == 'A' || charToCheck == 'a' || (charToCheck >= 'C' && charToCheck <= 'F') ||   
 (charToCheck >= 'c' && charToCheck <= 'f')) return ctHexadecimal;

if (charToCheck == 'b' || charToCheck == 'B') return ctBinaryMark;

if (charToCheck == 'o' || charToCheck == 'O') return ctOctalMark;

if (charToCheck == 'h' || charToCheck == 'H') return ctHexMark;

return ctUnknown;

}

bool checkString(const string& inputString) {

tState state = 1;

for (int i = 0; i < inputString.length(); ++i) {

state = transitions[state][getCharType(inputString[i])];

}

return isFinalState[state];

}

void findSubstrings(const string& inputString) {

tState state = 1;

bool hasSubstring = false;

int i, j;

for (int i = 0; i < inputString.length(); i++) {

state = 1;

for (j = i; j < inputString.length(); j++) {

state = transitions[state][getCharType(inputString[j])];

if (state == 0) break;

if (isFinalState[state]) {

cout << inputString.substr(i, j-i+1) << endl;

hasSubstring = true;

}

}

}

if (!hasSubstring) cout << "Корректных литералов не найдено!";

}

void printTask() {

cout << "Данная программа проверяет введенную строку на соответствие целочисленному   
 двоичному/восьмеричному/шестнадцатиричному литералу FASM в постфиксной форме.\n";

cout << "Для взаимодействия с программой используйте пукнты меню.\n";

}

void printMenu() {

cout << "\n0. Выйти из программы\n1. Проверить введенную строку на корректность;\n2. Выделить в   
 исходной строке все корректные целочисленные FASM литералы;\n3. Выделить в файле все   
 корректные целочисленные FASM литералы\n";

}

string readFile(string path) {

ifstream fin;

string buf;

fin.open(path);

fin >> buf;

fin.close();

return buf;

}

int main() {

setlocale(LC\_ALL, "Rus");

string literal;

bool isCorrect;

printTask();

int choice = 1;

while (choice) {

cout << "\nВыберите один из пунктов меню: ";

printMenu();

cin >> choice;

switch (choice) {

case 0:

break;

case 1:

cout << "Введите FASM литерал: ";

cin >> literal;

isCorrect = checkString(literal);

cout << ((isCorrect == true) ? "Это КОРРЕКТНЫЙ литерал!" : "Это НЕКОРРЕКТНЫЙ литерал!") <<   
 endl;

break;

case 2:

cout << "\nВведите строку для анализа: ";

getline(cin, literal);

getline(cin, literal);

findSubstrings(literal);

break;

case 3:

cout << "\nПуть к обрабатываемому файлу: F:/literal.txt\n";

literal = readFile("F:/literal.txt");

cout << "Исходная строка:" << literal << endl;

findSubstrings(literal);

break;

default:

cout << "Неверный ввод!\n";

break;

}

}

return 0;

}