**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕНЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.Шухова»**

**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 3

По дисциплине: Теория автоматов и формальных языков

Тема: «Регулярные языки и конечные распознаватели»

Выполнила: ст.группы ПВ - 31

Зановская А.И.

Проверил:

Рязанов Ю.Д.

Белгород

2018

**Цель работы:** изучить основные способы задания и построения регулярных языков, алгоритмы преобразования, анализа и реализации конечных распознавателей.

**Задание:**

1. Построить минимальный детерминированный конечный распознаватель заданного языка (см. варианты заданий).
2. Написать программу-распознаватель компиляционного и интерпретационного типа. Исходные данные: строка.

Результат:

* ***допустить*** – если строка представляет собой цепочку заданного языка;
* ***отвергнуть*** – в противном случае.

1. Написать программу, которая оставляет в исходном текстовом файле только те строки, которые представляют собой цепочки заданного языка.
2. Написать программу, которая исключает из исходного текстового файла строки, являющиеся цепочками заданного языка.

**Вариант №2:** язык строковых констант

Строковая константа – последовательность символов, которая заключена в кавычки. Последовательности символов «\n», «\t», «\b», «\r», «\f», «\’», «\”», «\\» и символ «\», за которым следует трехразрядное восьмеричное число, считаются одним символом, а одиночный символ «\» недопустим.

**Ход работы:**

Введем следующие обозначения:

* S – множество символов, включающее в себя различные буквы (за исключением символов «\» и «″»);
* C – символы преобразования «n», «t», «b», «r», «f», «’», «”», «\»;
* N – цифры в диапазоне от 0 до 7.

S, C

С

N

″

\

N

″

N

Исходный распознаватель, построенный по графу.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | ↓ |
|  | S0 | S1 | S2 | S3 | S4 | S5 |
| ″ | S1 | S5 |  |  |  |  |
| S |  | S1 |  |  |  |  |
| \ |  | S2 |  |  |  |  |
| C |  |  | S1 |  |  |  |
| N |  | S1 | S3 | S4 | S1 |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | K1 | | | | | | K2 |
|  | S0 | S1 | S2 | S3 | S4 |  | S5 |
| ″ | K1 | K2 | K1 | K1 | K1 | K1 | K1 |
| S | K1 | K1 | K1 | K1 | K1 | K1 | K1 |
| \ | K1 | K1 | K1 | K1 | K1 | K1 | K1 |
| C | K1 | K1 | K1 | K1 | K1 | K1 | K1 |
| N | K1 | K1 | K1 | K1 | K1 | K1 | K1 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | K1 | | | | | K2 | | K3 |
|  | S0 | S2 | S3 | S4 |  | | S1 | S5 |
| ″ | K2 | K1 | K1 | K1 | K1 | | K3 | K1 |
| S | K1 | K1 | K1 | K1 | K1 | | K2 | K1 |
| \ | K1 | K1 | K1 | K1 | K1 | | K1 | K1 |
| C | K1 | K2 | K1 | K1 | K1 | | K1 | K1 |
| N | K1 | K1 | K1 | K2 | K1 | | K2 | K1 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|  | S0 | S2 | S3 | S4 |  | S1 | S5 |
| ″ | K6 | K5 | K5 | K5 | K5 | K7 | K5 |
| S | K5 | K5 | K5 | K5 | K5 | K6 | K5 |
| \ | K5 | K5 | K5 | K5 | K5 | K2 | K5 |
| C | K5 | K6 | K5 | K5 | K5 | K5 | K5 |
| N | K5 | K3 | K4 | K6 | K5 | K6 | K5 |

bool IsLetter(char ch)

{

return ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'));

}

bool symb\_is\_S(char ch) { return (IsLetter(ch) && ch != '"'); }

bool symb\_is\_C(char ch) { return (ch == 'n' || ch == 't' || ch == 'b'

|| ch == 'r' || ch == 'f' || ch == '\'' || ch == '"' || ch == '\\'); }

bool symb\_is\_N(char ch) { return (ch >= '0' && ch <= '7'); }

bool recognizer\_compilation\_mode(std::string const &str)

{

unsigned \_condition = 0;

bool \_state = true;

for (unsigned i = 0; i < str.size(); i++)

{

switch (\_condition)

{

case 0: {

if (str[i] == '"') \_condition = 1;

else \_state = false;

} break;

case 1: {

if (symb\_is\_S(str[i])) \_condition = 1;

else if (str[i] == '"') \_condition = 5;

else if (str[i] == '\\') \_condition = 2;

else \_state = false;

} break;

case 2: {

if (symb\_is\_C(str[i])) \_condition = 1;

else if (symb\_is\_N(str[i])) \_condition = 3;

else \_state = false;

} break;

case 3: {

if (symb\_is\_N(str[i])) \_condition = 4;

else \_state = false;

} break;

case 4: {

if (symb\_is\_N(str[i])) \_condition = 1;

else \_state = false;

} break;

case 5: \_state = ((i + 1) == str.size());

}

if ((\_state == false) || (\_condition == 5)) break;

}

return \_state && (\_condition == 5);

}

void prepareForInterpret(int\*\*\* T, int \*m, int \*n)

{

int i, j;

FILE \*f = fopen("1.txt", "r");

fscanf\_s(f, "%i %i", &(\*m), &(\*n));

\*T = new int\*[\*m];

for (i = 0; i < \*m; ++i) {

(\*T)[i] = new int[\*n];

for (j = 0; j < \*n; ++j)

fscanf\_s(f, "%i", &((\*T)[i][j]));

}

}

int F(char x) {

if (x == '"') return 0;

if (symb\_is\_S(x)) return 1;

if (x == '\\') return 2;

if (symb\_is\_C(x)) return 3;

if (symb\_is\_N(x)) return 4; return -1;

}

bool recognizer\_interpretation\_mode(std::string const &str)

{

unsigned \_condition = 0;

bool \_state = true;

int n, m; int \*\*T;

int s = 0, spred = 0, k = 0;

int i = -1;

char x;

prepareForInterpret(&T, &m, &n);

do {

x = str[++i];//берем из строки s первый символ

s = T[F(x)][spred = s];

} while ((s >= 0) && (x != '\0'));

return s == 6;

}

unsigned input\_data(std::string const &fName, std::list<std::string> &dataList)

{

std::ifstream \_fin;

std::string \_buf;

\_fin.open(fName.c\_str());

unsigned \_n;

if (\_fin) {

\_fin >> \_n;

std::getline(\_fin, \_buf);

for (unsigned i = 0; i < \_n; i++) {

std::getline(\_fin, \_buf);

dataList.push\_back(\_buf);

}

\_fin.close();

}

else return 0;

return \_n;

}

int main()

{

std::list<std::string> \_dataList;

unsigned \_n;

if ((\_n = input\_data("data.txt", \_dataList)) != 0)

{

std::list<std::string>::const\_iterator \_ci = \_dataList.cbegin();

for (; \_ci != \_dataList.cend(); \_ci++)

{

std::cout << "========== next\_string" << std::endl << "string: " << std::string(\*\_ci) << std::endl

<< "status: " << (recognizer\_compilation\_mode(\*\_ci) ? "TRUE" : "FALSE") << std::endl;

/\* std::cout << "========== next\_string" << std::endl << "string: " << std::string(\*\_ci) << std::endl

<< "status: " << (recognizer\_interpretation\_mode(\*\_ci) ? "TRUE" : "FALSE") << std::endl;\*/

}

}

else std::wcout << L"### ERR: 1 (main)" << std::endl;

return 0;

}

