ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

ассистент		Д.А. Кочин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5

«СИНТЕЗ КОНЕЧНЫХ АВТОМАТОВ»

по курсу: ТЕОРИЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4031		Х.В. Сидиропуло
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

Цель работы:

В данной лабораторной работе требуется:

- Построить конечный автомат, который осуществляет проверку входного слова на допустимость в заданном регулярном выражении используя алгоритм синтеза конечных автоматов;
- Привести в отчете процесс синтеза конечного автомата;
- Создать программу на языке высокого уровня реализующую алгоритм синтеза конечного автомата на основе заданного регулярного выражения.

Требования к программе:

Входными данными является текстовый файл, содержащий регулярное выражение. Выходными данными является текстовый файл, содержащий автоматную матрицу построенного КНА.

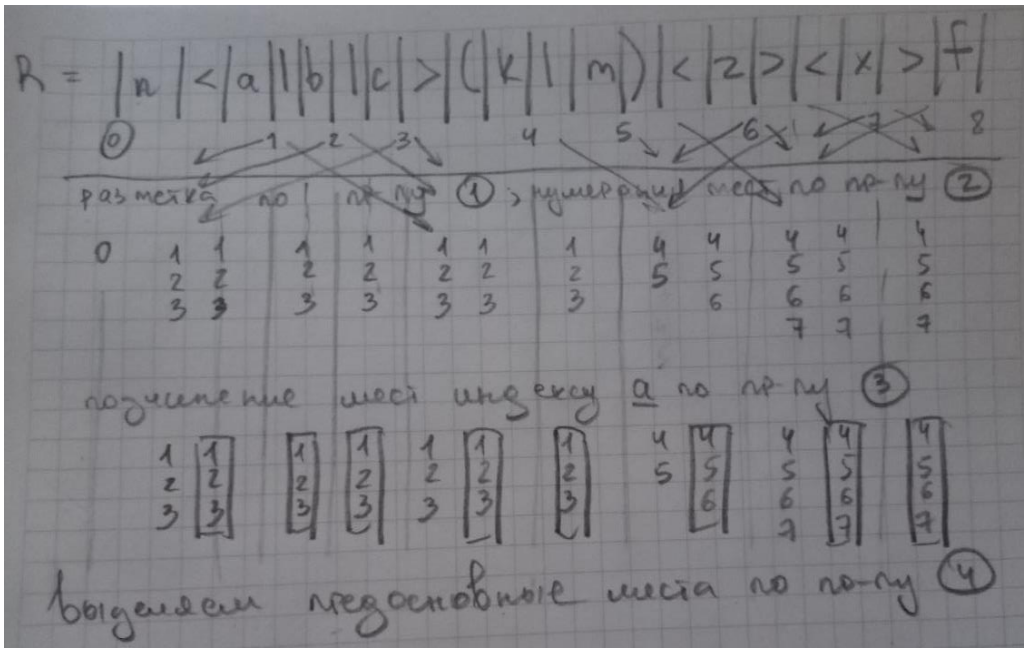
Вариант 17: n<a|b|c>(k|m)<z><x>f

S = < X, Q, U, δ, λ>

X = {a, b, c, k, m, n, f, z, x} — алфавит входных символов

Q = {q0, q1, q2, q3, q4, q5, q6, q7, q8, q9} — множество состояний

U = {0, 1} — алфавит выходных символов



Алгоритм синтеза КНА:

Описание алгоритма синтеза КНА

Примем следующее правило отметки состояний КНА:

- Состояние qi КНА, определяемое подмножеством множества основных мест регулярных выражений R1,…,Rp, отмечается множеством, содержащим все те и только те выражения R1,…,Rp, конечные места которых подчинены хотя бы одному основному месту из числа мест, входящих в подмножество для qi

Из предыдущего изложения следует следующее утверждение: Слово a в алфавите X регулярных выражений тогда и только тогда переводит КНА из начального состояния q1 в состояние qj, отмеченное произвольным множеством, содержащим любое из заданных рег. Выражений Ri, тогда начальное место выражения Ri связано с конечным местом этого выражения словом a.

Пусть X – алфавит для R (без «(», «)», «|», «&», «<», «>»)

Основным местом в R назовем место, непосредственно слева от которого стоит символ (буква) алфавита X, а также начальное место.

Место непосредственно справа от которого стоит буква алфавита X, будем называть пред основным местом

Правила подчинения мест в регулярном выражении

1) Начальные места всех термов (букв) многочлена, помещенного в обычные или итерационные скобки, подчинены месту, расположенному непосредственно слева от открывающей скобки

2) Место расположенное непосредственно справа от закрывающей скобки, подчинено конечным местам всех термов многочлена, заключенного в эти скобки; а в случае итерационных скобок – еще и месту, расположенному непосредственно слева от соответствующей открывающей скобки.

3) Начальные места всех термов многочлена, заключенного в итерационные скобки, подчинены месту, расположенному непосредственно справа от соответствующей закрывающей скобки.

4) Если место c подчинено месту b, а место b подчинено месту a, то место c подчинено месту a.

5) Каждое место подчинено самому себе

6) Других случаев подчинения мест в R нет.

Разметка регулярных выражений

R – регулярное выражение в алфавите X={x1,x2,…,xn}

Разделяющими местами будем называть специальные знаки («|») – вертикальные черточки, стоящие между символами в R/

Среди них будем выделять начальное и конечное места

Пусть R – произвольное регулярное выражение, тогда

1) Место a связано словом “альфа” с местом b в R, если от места a к месту b можно перейти с помощью любого числа

- Непосредственных переходов (переход через E-пустой символ)
- Переход через символы xi1, xi2, xin, взятые по одному разу в том порядке, в каком они входят в слово «альфа»

2) Место b альфа-следует за местом a тогда, когда место a связано с место b словом альфа

3) Место b подчинено месту a, если от места a к месту b можно перейти с помощью одних лишь непосредственных переходов, т.е. если место a связано с местом b пустым словом E

Пример:

- Непосредственный переход от места 0 к месту 4
- Переход через Z от места 1 к месту 2

Ход работы:

Q/x	q0	q1	q2	q3	q4	q5	q6	q7	q8	q9
n	q1/0									
a		q2/0	q2/0	q2/0	q2/0					
b		q3/0	q3/0	q3/0	q3/0					
c		q4/0	q4/0	q4/0	q4/0					
k		q5/0	q5/0	q5/0	q5/0					
m		q6/0	q6/0	q6/0	q6/0					
z						q7/0	q7/0	q7/0		
x						q8/0	q8/0	q8/0	q8/0	
f						q9/1	q9/1	q9/1	q9/1	

Таблица 1 — КНА в виде матрицы переходов/выходов

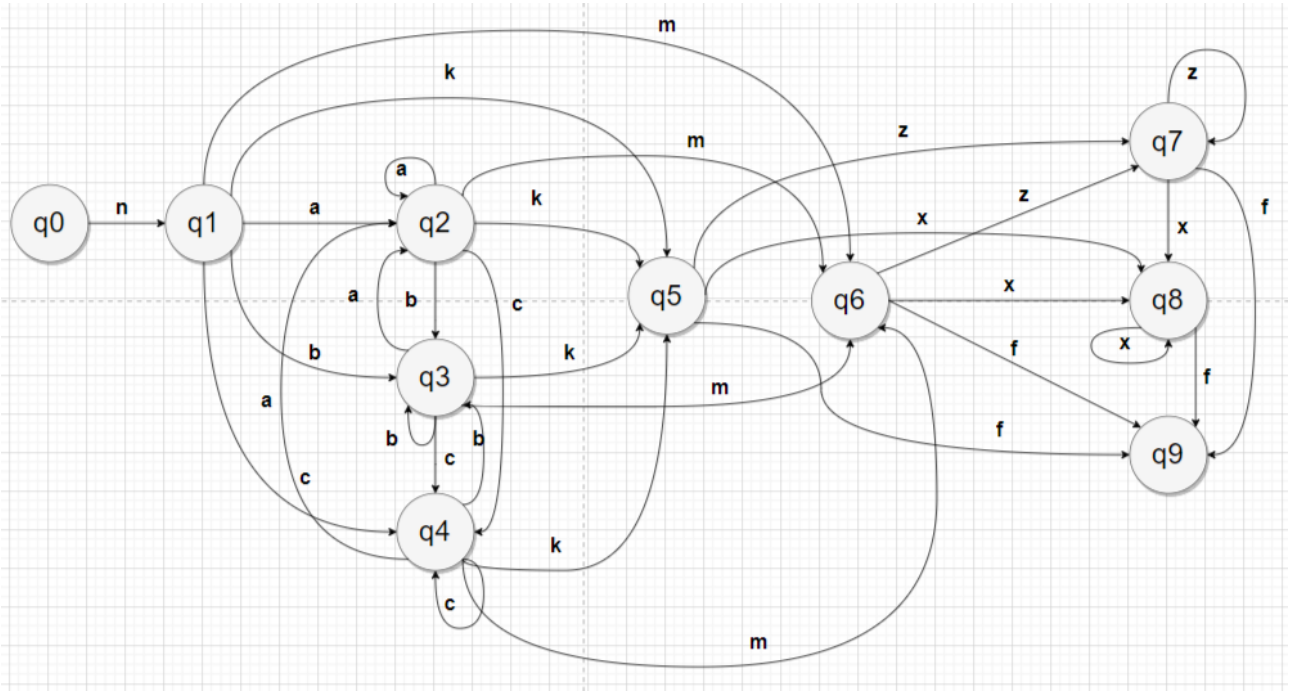


Рис.1 — КНА в виде графа переходов

Q \ Q	q0	q1	q2	q3	q4	q5	q6	q7	q8	q9
q0		n/0								
q1			a/0	b/0	c/0	k/0	m/0			
q2			a/0	b/0	c/0	k/0	m/0			
q3			a/0	b/0	c/0	k/0	m/0			
q4			a/0	b/0	c/0	k/0	m/0			
q5								z/0	x/0	f/1
q6								z/0	x/0	f/1
q7								z/0	x/0	f/1
q8									x/0	f/1
q9										

Таблица 2 — КНА в виде автоматной матрицы

Упрощаем:

Q/x	q0	q1	q2	q3	q4	q5	q6	q7	q8	q9
n	q1/0									
a		q2/0	q2/0	q2/0	q2/0					
b		q3/0	q3/0	q3/0	q3/0					
c		q4/0	q4/0	q4/0	q4/0					
k		q5/0	q5/0	q5/0	q5/0					
m		q6/0	q6/0	q6/0	q6/0					
z						q7/0	q7/0	q7/0		
x						q8/0	q8/0	q8/0	q8/0	
f						q9/1	q9/1	q9/1	q9/1	
Класс	**0**	**1**	**1**	**1**	**1**	**2**	**2**	**2**	**3**	**4**

Таблица 3 — КНА в виде матрицы переходов/выходов

Q	Класс		Q	Класс		
0	Q11		0	Q21		$q0^0$
1,2,3,4	Q12	=>	1,2,3,4	Q22	=>	$q1^0$
5,6,7	Q13		5,6,7	Q23		$q2^0$
8	Q14		8	Q24		$q3^0$
9	Q15		9	Q25		$q4^0$

Таблица 4 — минимизация КНА

Как видно из шагов минимизации, количество состояний не изменилось, следовательно, останавливаемся и можем строить минимизированный КНА.

Q^0/x	n	a	b	c	k	m	z	x	f
$q0^0$	q1/0								
$q1^0$		q1/0	q1/0	q1/0	q2/0	q2/0			
$q2^0$							q2/0	q3/0	q4/1
$q3^0$								q3/0	q4/1
$q4^0$									

Таблица 5 — минимизированный КНА в виде матрицы переходов/выходов

Q^0/x	$q0^0$	$q1^0$	$q2^0$	$q3^0$	$q4^0$
$q0^0$		n/0			
$q1^0$		a,b,c/0	k,m/0		
$q2^0$			z/0	x/0	f/1
$q3^0$				x/0	f/1
$q4^0$					

Таблица 6 — КНА в виде автоматной матрицы

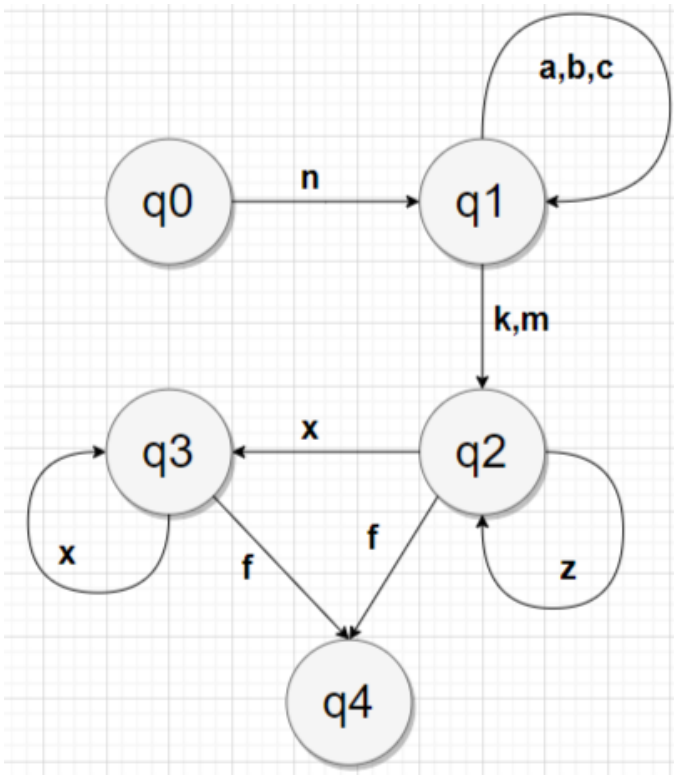
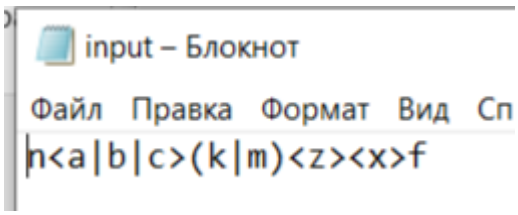


Рис.2 — минимизированный КНА в виде графа переходов

Результат работы программы:

Входной файл:



Результирующий файл:

result – Блокнот

Файл Правка Формат Вид Справка

	0	1	2	3	4	5	6	7	8	9
n	1/0	-	-	-	-	-	-	-	-	-
a	-	2/0	2/0	2/0	2/0	-	-	-	-	-
b	-	3/0	3/0	3/0	3/0	-	-	-	-	-
c	-	4/0	4/0	4/0	4/0	-	-	-	-	-
k	-	5/0	5/0	5/0	5/0	-	-	-	-	-
m	-	6/0	6/0	6/0	6/0	-	-	-	-	-
z	-	-	-	-	-	7/0	7/0	7/0	-	-
x	-	-	-	-	-	8/0	8/0	8/0	8/0	-
f	-	-	-	-	-	9/1	9/1	9/1	9/1	-

Листинг программы:

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <algorithm>
using namespace std;
struct row//заголовок строки таблицы 4 шага. Содержит для облегчения дальнейших шагов
позиции символа
{
	string symbol;
	vector <int> place;
};

void r_file(string& st, string name);
void razmetka(vector <vector <int>>& razdelitel, string str, int begin, int end, int& count,
vector<row>& r);
void dop(vector <vector <int>>& razdelitel, string str, int begin, int end);
void punkt4(vector<vector<vector <int>>>& cell, vector<row>rOfTable, vector<vector <int>>&
cOfTable, vector <vector <int>> razdelitel, vector<bool>& correctly);
void w_file(vector<vector<vector <int>>> cell, vector<row>rOfTable, vector<vector
<int>>cOfTable, vector<bool>correctly);

int zScobka(string str, int begin);
int find(vector<row> r, char s[1]);
int main()

{
	setlocale(LC_ALL, "rus");
	string str;
	r_file(str, "input.txt");
	vector <vector <int>> razdelitel(str.size() + 1, vector<int>());
	int count = 1;
	vector<row>rOfTable;
	vector<bool>correctly;
	vector<vector <int>>cOfTable;//заголовки столбцов таблицы
	cOfTable.push_back(vector < int > {0});
	correctly.push_back(false);
	razmetka(razdelitel, str, 0, str.size(), count, rOfTable);
	vector<vector<vector <int>>> cell(rOfTable.size(), vector<vector <int>>());//первый
вектор-строка, второй-столбец, третий - ячейка
	punkt4(cell, rOfTable, cOfTable, razdelitel, correctly);
	w_file(cell, rOfTable, cOfTable, correctly);
	system("pause");
	return 0;
}
void w_file(vector<vector<vector <int>>> cell, vector<row>rOfTable, vector<vector
<int>>cOfTable, vector<bool>correctly)

{
	ofstream file("result.txt");
	vector<vector<int>>::iterator it;
	for (int i = 0; i < cOfTable.size(); i++)
	{
		file << '\t' << i;
	}
	for (int i = 0; i < rOfTable.size(); i++)
	{
		file << '\n' << rOfTable[i].symbol << '\t';
		for (int j = 0; j < cOfTable.size(); j++)
		{
			if (cell[i][j].empty())
				file << "-\t";
			else
```

```
                        {
                                it = find(cOfTable.begin(), cOfTable.end(), cell[i][j]);
                                file << it - cOfTable.begin() << '/';
                                if (correctly[it - cOfTable.begin()] == true)
                                        file << '1';
                                else
                                        file << '0';
                                file << '\t';
                        }
                }
        }
        file.close();
}
void punkt4(vector<vector<vector <int>>>& cell, vector<row>rOfTable, vector<vector <int>>&
cOfTable, vector <vector <int>> razdelitel, vector<bool>& correctly)
{
        int column = 0;
        while (column < cOfTable.size())
        {
                for (int i = 0; i < cOfTable[column].size(); i++)//состояние является
конечным?
                {
                        for (int j = 0; j < razdelitel[razdelitel.size() - 1].size(); j++)
                        {
                                if (cOfTable[column][i] == razdelitel[razdelitel.size() - 1][j])
                                {
                                        correctly[column] = true;
                                        break;
                                }
                        }
                        if (correctly[column] == true)
                                break;
                }
                for (int i = 0; i < rOfTable.size(); i++)//каждая строка
                {
                        cell[i].push_back(vector<int>());
                        for (int j = 0; j < rOfTable[i].place.size(); j++)//для каждого
вхождения символа во входную строку
                        {
                                for (int k = 0; k < cOfTable[column].size(); k++)//дл€ каждой
позиции в заголовке столбца
                                {
                                        if
(find(razdelitel[rOfTable[i].place[j]].begin()/*предъосновное место этого символа i в месте
j*/, razdelitel[rOfTable[i].place[j]].end(), cOfTable[column][k]) !=
razdelitel[rOfTable[i].place[j]].end()
                                                //в этом предъосновном месте найдена позиция из
заголовка столбца
                                                && find(cell[i][column].begin(),
cell[i][column].end(), razdelitel[rOfTable[i].place[j] + 1][0]) == cell[i][column].end())
                                                //и позиции основного места нет в ячейки

                        cell[i][column].push_back(razdelitel[rOfTable[i].place[j] + 1][0]); //значит
добавл€ем позицию основного места в ячейку
                                }
                        }
                        sort(cell[i][column].begin(), cell[i][column].end());
                        if (find(cOfTable.begin(), cOfTable.end(), cell[i][column]) ==
cOfTable.end() && !cell[i][column].empty())//если такого заголовка нет
                        {
                                cOfTable.push_back(cell[i][column]);//добавляем
                                correctly.push_back(false);
                        }
                }
                column++;
```

```
	}
}
void razmetka(vector <vector <int>>& razdelitel, string str, int begin, int end, int& count,
vector<row>& r)// count = 1
{
	vector<int>dMemberB;
	vector<int>dMemberE;
	int check;
	for (int i = begin; i < end; i++)
	{
		if (i == begin)
		{
			if (i == 0)
			{
				razdelitel[0].push_back(0);
				if (str[0] != '<' && str[0] != '>' && str[0] != '(' && str[0] !=
')')
					dMemberB.push_back(0);
			}
			else
			{
				for (int j = 0; j < razdelitel[i - 1].size(); j++)
					razdelitel[i].push_back(razdelitel[i - 1][j]);
				dMemberB.push_back(i);//начальные места дизъюнктивных членов
			}
		}
		if (str[i] != '<' && str[i] != '>' && str[i] != '(' && str[i] != ')' && str[i]
!= '|')//отмечаю основные места
		{
			check = find(r, &str[i]);
			if (check == -1)
				r.push_back(row{ { str[i] },{ i } });
			else
				r[check].place.push_back(i);
			razdelitel[i + 1].push_back(count);
			count++;
		}
		if (i > 0)
		{
			if (str[i] != '>' && str[i] != ')' && str[i] != '|' && str[i - 1] ==
'|')//Индекс места перед любыми скобками распростран€етс€ на начальные места
			//всех дизъюнктивных членов, записанных в этих скобках
			{
				if (begin == 0)
				{
					for (int j = 0; j < razdelitel[begin].size(); j++)
						razdelitel[i].push_back(razdelitel[begin][j]);
				}
				else
				{
					for (int j = 0; j < razdelitel[begin - 1].size(); j++)
						razdelitel[i].push_back(razdelitel[begin - 1][j]);
				}
				dMemberB.push_back(i);//начальные места дизъюнктивных членов
				dMemberE.push_back(i - 1);
			}
		}
		if (i == end - 1)
		{
			if (str[end - 1] == ')' || str[end - 1] == '>')//отмечаю конец
последнего диз члена в скобках
				dMemberE.push_back(end - 1);
			for (int j = 0; j < dMemberE.size(); j++)//Индекс конечного места
любого дизъюнктивного члена, заключенного в любые скобки,
```

```
                        //распространяется на место, непосредственно следующее за этими
скобками.
                        {
                                for (int k = 0; k < razdelitel[dMemberE[j]].size(); k++)
                                        razdelitel[end].push_back(razdelitel[dMemberE[j]][k]);
                        }
                        if (str[end - 1] == '>')
                        {
                                for (int j = 0; j < razdelitel[begin - 1].size(); j++)
                                        //индекс места перед итерационными скобками
распространяется на место, непосред - ственно следующее за этими скобками
                                        if (find(razdelitel[end].begin(), razdelitel[end].end(),
razdelitel[begin - 1][j]) == razdelitel[end].end())
                                                razdelitel[end].push_back(razdelitel[begin -
1][j]);
                                for (int j = 0; j < dMemberB.size(); j++)
                                        //индекс места за итерационными скобками распространяется
на начальные места всех дизъюнктивных членов, заключенных в итерационные скобки
                                {
                                        for (int k = 0; k < razdelitel[end].size(); k++)
                                        {
                                                if (find(razdelitel[dMemberB[j]].begin(),
razdelitel[dMemberB[j]].end(), razdelitel[end][k]) == razdelitel[dMemberB[j]].end())

        razdelitel[dMemberB[j]].push_back(razdelitel[end][k]);
                                        }
                                        if (str[dMemberB[j]] == '<' || str[dMemberB[j]] ==
'('))//значит надо добавить на начальные места диз членов новые индексы
                                        {
                                                dop(razdelitel, str, dMemberB[j] + 1, zScobka(str,
dMemberB[j]));
                                        }
                                }
                        }
                }
                if (str[i] == '<' || str[i] == '(')
                {
                        int zSc = zScobka(str, i);
                        razmetka(razdelitel, str, i + 1, zSc + 1, count, r);
                        i = zSc;
                }
        }
}
void dop(vector <vector <int>>& razdelitel, string str, int begin, int end)
{
        for (int i = begin; i < end; i++)
        {
                if (i == begin)//добавляем в начальное место, сразу за скобками
                {
                        for (int j = 0; j < razdelitel[begin - 1].size(); j++)
                        {
                                if (find(razdelitel[begin].begin(), razdelitel[begin].end(),
razdelitel[begin - 1][j]) == razdelitel[begin].end())
                                        razdelitel[begin].push_back(razdelitel[begin - 1][j]);
                        }
                }
                if (str[i] != '>' && str[i] != ')' && str[i] != '|' && str[i - 1] == '|')//
добавляем всем остальным диз членам в скобках
                {
                        for (int j = 0; j < razdelitel[begin - 1].size(); j++)
                        {
                                if (find(razdelitel[i].begin(), razdelitel[i].end(),
razdelitel[begin - 1][j]) == razdelitel[i].end())

                                        razdelitel[i].push_back(razdelitel[begin - 1][j]);
```

```
                        }
                }
                if ((str[i] == '<' || str[i] == '(') && (str[i - 1] == '|' || str[i - 1] ==
'(' || str[i - 1] == '<'))
                {
                        int zSc = zScobka(str, i);
                        dop(razdelitel, str, i + 1, zSc);
                        i = zSc;
                }
        }
}
int find(vector<row> r, char s[1])
{
        for (int j = 0; j < r.size(); j++)
        {
                if (s[0] == r[j].symbol[0])
                        return j;
        }
        return -1;
}

int zScobka(string str, int begin)
{
        int count = 1;
        string scobki;
        if (str[begin] == '<')
                scobki = "<>";
        else
                scobki = "()";
        while (1)
        {
                begin++;
                if (str[begin] == scobki[0])
                        count++;
                if (str[begin] == scobki[1])
                {
                        count--;
                        if (count == 0)
                                return begin;
                }
        }
}

void r_file(string& st, string name)
{
        string temporary;
        ifstream file(name);
        if (!file.is_open())
                cout << "Файл не может быть открыт!\n";
        else
        {
                string str;
                getline(file, str);
                if (str.empty()) cout << "Файл пуст!" << endl;
                else

                {
                        st = str;
                }
        }
        file.close();
        return;
}
```

Вывод:

В результате выполнения лабораторной работы были рассмотрены основные понятия теории КНА, создана программа на языке высокого уровня, реализующую алгоритм синтеза конечного автомата на основе заданного регулярного выражения.