# Работа

# Задание на самостоятельную работу «Технологии и методы программирования»

### Тема задания “Технологии программирования прикладных экономических систем”

* Целью выполнения самостоятельной работы является разработка встроенного языка и программного кода, для обеспечения лексического разбора.
* В процессе выполнения самостоятельной работы требуется:
* Во-первых, самостоятельно выбрать предметную область прикладного программного средства;
* Во-вторых, описать предметную область с использованием теории графов. Построить граф предметной области;
* В-третьих, разработать программный код, включающий лексический разбор программы на встроенном языке, с использованием детерминированного конечного автомата. Программный код может быть реализован в любой системе программирования;
* В-четвертых, подготовить отладочные варианты программы на разработанном встроенном языке прикладного программного средства. Отладочные варианты должны подтверждать, как разбор корректной программы, так и обнаружение ошибок в программе на встроенном языке.

## Оформление самостоятельной работы

### Отчет должен включать следующие элементы:

1. Титульный лист.
2. Оглавление.
3. 3.Описание предметной области с использованием теории графов (граф предметной области)
4. Описание грамматики встроенного языка
5. Таблица детерминированного конечного автомата.
6. Распечатка программного кода.
7. 7.Распечатки экранов, подтверждающих выполнение.
8. 8. Литература и INTERNET-ресурсы.

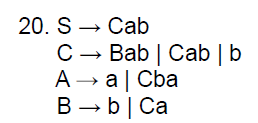
# Работа

**Выполнение**

**Конструирование и программное моделирование конечного автомата**

Целью работы является получение навыков разработки регулярных языков и конструирования конечных автоматов, используемых в качестве моделей простейших элементов транслятора.

Преобразуем регулярную грамматику G{{a,b}, {S, C, B, A}, P, {S}} к автоматному виду и построим по ней конечный автомат, постройте граф переходов для данного автомата.



Решение:

Построим множество VN’ = {S, C, A, B}.

Правила S → Cab и C → Bab | Cab | b меняем на S → Bab | Cab | b. Для правила S→Bab во множество VN’ включаем символ С1, а само правило разбиваем на два: S→C1b, C1→Ba, и включаем эти правила во множество P’.

Для правила S→Cab во множество VN’ включаем символ С2, а само правило разбиваем на два: S→C2b, C2→Sa, и включаем эти правила во множество P’.

Правило S→b оставляем без изменений.

Так же меняем правила A → Cba и B → Ca на A → Sba и B → Sa. Правила A→a и B→b оставляем без изменений. Для правила A→ Sab во множество VN’ включаем символ C3, а само правило разбиваем на два: A → С3b и С3 → Sa и включаем эти правила во множество P’.

G’({a, b], {S, C1, C2, C3, A, B}, P, {S})

P’:

S → C1b | Ba | C2b

C1 → Ba

C2 → Sb

C3 → Sa

A → a | C3b

B → b | Sa

Построение конечного автомата M (Q, V, δ, q0, F):

Строим множество состояний автомата Q = VB  {H} = {S, C1, C2, C3, A, B, H}, алфавит входных символов V = {a, b}

Функции переходов конечного автомата:

δ(b, C1)=S; δ(a, B)=S; δ(b, C2)=S; δ(a, B)=C1; δ(b, S)=C2; δ(a, D)=C3; δ(a, H)=A;

δ(b, C3)=A; δ(b, H)=B; δ(a, S)=B; δ(a, S)=C2;

Начальное состояние q0=H;

Множество конечных состояний F = {A}

Конечный автомат принимает следующий вид:

M({S, C1, C2, C3, A, B, H}, {a, b}, δ, H, {A})

Приводим конечный автомат к определенному виду и добавляем ошибку Е. Функции переходов: δE(a, C1) = E; δE(b, B) = E; δE(b, C2) = E;

Представим автомат в виде графа:

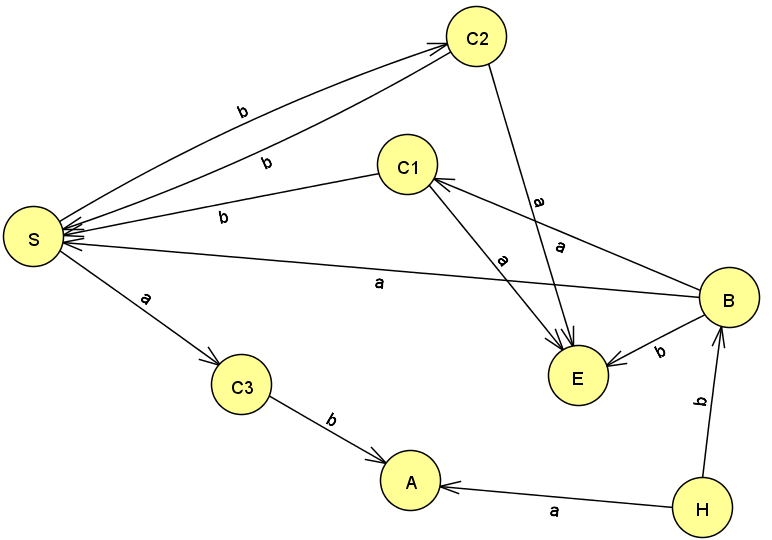


Рис.1

Недетерминированный конечный автомат (НКА) - это пятерка M = (Q, T, D, q0, F), где

* Q - конечное множество состояний;
* T - конечное множество допустимых входных символов (входной алфавит);
* D - функция переходов (отображающая множество Q(Tcmsy10-5b{e}) во множество подмножеств множества Q), определяющая поведение управляющего устройства;
* q0 cmsy10-32Q - начальное состояние управляющего устройства;
* F cmsy10-12Q - множество заключительных состояний.

Работа конечного автомата представляет собой некоторую последовательность шагов, или тактов. Такт определяется текущим состоянием управляющего устройства и входным символом, обозреваемым в данный момент входной головкой. Сам шаг состоит из изменения состояния и, возможно, сдвига входной головки на одну ячейку вправо (рис. 2).

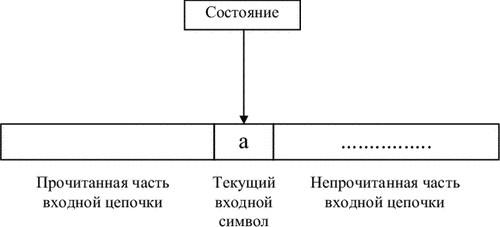


Рис.2

Недетерминизм автомата заключается в том, что, во-первых, находясь в некотором состоянии и обозревая текущий символ, автомат может перейти в одно из, вообще говоря, нескольких возможных состояний, и во-вторых, автомат может делать переходы по e.

Важным частным случаем недетерминированного конечного автомата является детерминированный конечный автомат, который на каждом такте работы имеет возможность перейти не более чем в одно состояние, и не может делать переходы по e.

Конечный автомат – это модель устройства с конечным числом состояний, которое отличает правильно образованные, или «допустимые» цепочки, от «недопустимых».

# Составление формальной грамматики.

Фраза языка представляет собой список, поэтому из начального символа грамматики должен выводиться список:

R0: <предложение> ::==< фраза>#

R1: <фраза> ::==< фраза>/< составное имя> | <идентификатор>

R2: <составное имя> ::==<составное имя>.<идентификатор>| <идентификатор>

R3: <идентификатор>::==<идентификатор><допустимый символ>| <начальный символ>

R4: <начальный символ>::==<буква>|<знак подчёркивания>

R5: <допустимый символ> :=:=<буква>|<цифра>|<знак подчёркивания>

R6:<буква>::==a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|

R7: <цифра>::==0|1|2|3|4|5|6|7|8|9|

R8:<знак подчёркивания>::==\_

Таким образом, требуемую грамматику можно описать следующей структурой:

* Множество терминальных символов: 0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,g,h,I,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,\_, .,/,#
* Множество нетерминальных символов: <фраза>, < составное имя>, <идентификатор>, <допустимый символ>, <начальный символ>, <буква>, <цифра>, <знак подчёркивания>.
* Множество правил вывода R0,R1, R2, R3, R4, R5, R6, R7, R8.

# Построение конечного автомата.

Между конечными автоматами и автоматными грамматиками существует тесная связь: класс языков, допускаемых конечными автоматами, совпадает с классом языков, порождаемых автоматными грамматиками.

Для построения конечного автомата составленную грамматику путем введения дополнительных состояний надо преобразовать к автоматному виду, в результате получится следующая таблица переходов:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | \_ | . | / | # |
| Да | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Нет | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Составное  имя | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 |
| Идентификатор | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 2 | 1 |
| Начальный  символ | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 8 | 5 | 2 | 1 |
| Допустимый  символ | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 8 | 5 | 1 | 1 |
| Буква | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 8 | 5 | 2 | 0 |
| Цифра | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 8 | 5 | 2 | 0 |
| Знак  подчёркивания | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 8 | 5 | 2 | 0 |

# Программное моделирование работы

# конечного автомата.

Листинг программного кода для моделирования работы конечного автомата имеет следующий вид:

#include "iostream.h"

#include "stdio.h"

#include "conio.h"

int main()

{ int i,j,kol,tsost,slsost,tsymb;

int tabl[9][40]={

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},//da {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},//net {1,1,1,1,1,1,1,1,1,1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,1,1,1},//sost imya {7,7,7,7,7,7,7,7,7,7,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,5,2,1},//ident {7,7,7,7,7,7,7,7,7,7,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,8,5,2,1},//nach sim {7,7,7,7,7,7,7,7,7,7,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,8,5,1,1},//dop sim {7,7,7,7,7,7,7,7,7,7,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,8,5,2,0},//bykva {7,7,7,7,7,7,7,7,7,7,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,8,5,2,0},//cifra {7,7,7,7,7,7,7,7,7,7,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,8,5,2,0}//znak

};printf("matrica\n");

for (i=0;i<9;i++) {for (j=0;j<40;j++) printf("%4d",tabl[i][j]); printf("\n");};

char ch, inpstr[80] ;

printf("\n ENTER STRING ");

i=0;

while ((ch=getch()) !=13 && i<80)

{putch(ch);

inpstr[i++]=ch;}

inpstr[i]='\0';

kol=i-1;

printf("\n input string:");

printf(inpstr);

printf("\n");

tsost=2;

for (i=0;i<=kol;i=i+1)

{ tsymb=inpstr[i];

switch (tsymb)

{ case '0': slsost=tabl[tsost][0]; break;

case '1': slsost=tabl[tsost][1]; break;

case '2': slsost=tabl[tsost][2]; break;

case '3': slsost=tabl[tsost][3]; break;

case '4': slsost=tabl[tsost][4]; break;

case '5': slsost=tabl[tsost][5]; break;

case '6': slsost=tabl[tsost][6]; break;

case '7': slsost=tabl[tsost][7]; break;

case '8': slsost=tabl[tsost][8]; break;

case '9': slsost=tabl[tsost][9]; break;

case '/': slsost=tabl[tsost][10]; break;

case '.': slsost=tabl[tsost][11]; break;

case 'e': slsost=tabl[tsost][12]; break;

case '+': slsost=tabl[tsost][13]; break;

case '-': slsost=tabl[tsost][14]; break;

case '#': slsost=tabl[tsost][15]; break;

default: slsost=1;}

printf("%5d",slsost);

tsost=slsost;

};

switch (slsost)

{ case 1:cout<<"\n STRING is WRONG \n"; break;

case 0:cout<<"\n STRING is RIGHT \n";break;

case 2:cout << "\n STRING is WRONG \n";break;

}

return 0;

};

результат выполнения программы:



Входной текстовый файл matr.txt

9 количество состояний

40 количество символов алфавита

9

40

0 1 2 3 4 5 6 7 8 9 a b c d e f g h i j k l m n o p q r s t u v w x y z \_ . / #

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1

5 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 2 1

7 7 7 7 7 7 7 7 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 8 5 2 1

7 7 7 7 7 7 7 7 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 8 5 1 1

7 7 7 7 7 7 7 7 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 8 5 2 0

7 7 7 7 7 7 7 7 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 8 5 2 0

7 7 7 7 7 7 7 7 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 8 5 2 0

2

3

0

0

0

Заголовочный файл konavt.h

class konavt {int kolsost;// число состояний автомата

int kolsimv;// число символов входного алфавита

char \*alfavit; // входной алфавит

int nachstate; //начальное состояние

int kolkon;//число заключительных состояний

int \*fin; //заключительные состояния

int \*\*per;//матрица переходов

int dlina;// текущая длина входной цепочки

int dlina0;// начальная длина входной цепочки

char \*vxod;// входная цепочка

int avtstate; //текущее состояние

int nomstep;// номер шага автомата

int \*protokol;// протокол работы автомата

public: bool error; //признак ошибки

konavt();//конструктор без параметров

void show\_sost();//проверка заполнения матрицы

void sled();//срабатывание переходов

void show();//показ текущего состояния;

void init();//инициализация новой строкой

bool konec();// проверка финальности состояния

bool zaversh(); //проверка исчерпания строки

bool proverka();// проверка принадлежности символов алфавиту

};

Реализация класса konavt.cpp

# include "konavt.h"

# include <iostream.h>

# include <fstream.h>

# include <string.h>

# include <stdlib.h>

//конструктор

konavt::konavt()

{int i,j,vyb;

// возможен ввод исходных данных как из файла, так и с клавиатуры

cout<<"constructor working..."<<endl;

cout <<"Istochnik dannyx(0-klaviatura,1-file)"<<endl;

cin>>vyb;

// ввод с клавиатуры

if (vyb==0)

{ cout<<"\n Enter kolvo sostoianiy\t";

cin>>kolsost;

cout<<endl<<"enter kolvo simvolov alfavita\t";

cin>>kolsimv;

alfavit=new char [kolsimv];

per=new int\*[kolsost];

for (i=0;i<kolsost;i++){

per[i]=new int [kolsimv];

};

cout<<endl<<"enter alfavit"<<endl;

for (j=0;j<kolsimv;j++){

cin>>alfavit[j];

};

cout<<endl<<"enter matrica"<<endl;

for (i=0;i<kolsost;i++){

for (j=0;j<kolsimv;j++){

cin>>per[i][j];

};

cout<<endl;

};

cout<<"enter nachalnoe sostoianie"<<endl;

cin>>nachstate;

cout<<endl;

cout<<"enter kolvo konechnyh sostoianiy"<<endl;

cin>>kolkon;

cout<<endl;

fin=new int[kolkon];

cout<<"enter konechnye sostoiania"<<endl;

for (i=0;i<kolkon;i++){

cin>>fin[i];

}

cout<<endl;

// запись исходных данных в файл

int otv;

cout<<"save to file?(1-yes,0-no)";

cin>>otv;

cout<<endl;

if (otv==1)

{char fname[30];

cout<<"enter filename ";

cin>>fname;

cout<<endl;

ofstream out\_stream;

out\_stream.open(fname);

if (out\_stream.fail())

{cout<<"Error output file"<<endl; return;}

out\_stream<<kolsost<<' ';

out\_stream<<kolsimv<<' ';

for (i=0;i<kolsimv;i++) out\_stream<<alfavit[i]<<' ';

for (i=0;i<kolsost;i++){

for (j=0;j<kolsimv;j++){

out\_stream<<per[i][j]<<' ';}}

out\_stream<<nachstate<<' ';

out\_stream<<kolkon<<' ';

for (i=0;i<kolkon;i++)out\_stream<<fin[i]<<' ';

out\_stream.close();

cout<<"End of output file..."<<endl;

};

} else

// ввод исходных данных из файла

{ char filename[30];

cout<<"Enter Filename ";

cin>>filename;

cout<<endl<<"vvedeno "<<filename<<endl;

ifstream in\_stream;

in\_stream.open(filename);

if (in\_stream.fail())

{cout<<"net faila "<<filename<<endl; return;

};

in\_stream>>kolsost;

cout<<"kolsost="<<kolsost<<endl;

in\_stream>>kolsimv;

cout<<"kolsimv="<<kolsimv<<endl;

alfavit=new char[kolsimv];

for (i=0;i<kolsimv;i++){ in\_stream>>alfavit[i];};

for (i=0;i<kolsimv;i++) cout<<alfavit[i]<<" ";

cout<<endl;

per=new int\*[kolsost];

for (i=0;i<kolsost;i++) per[i]=new int [kolsimv];

for (i=0;i<kolsost;i++){

for (j=0;j<kolsimv;j++){

in\_stream>>per[i][j];}}

for (i=0;i<kolsost;i++){

for (j=0;j<kolsimv;j++){

cout<<per[i][j];}cout<<endl;}

in\_stream>>nachstate;

cout<<"begin state "<<nachstate<<endl;

in\_stream>>kolkon;

cout<<"Number of end states "<<kolkon<<endl;

fin=new int[kolkon];

for (i=0;i<kolkon;i++)in\_stream>>fin[i];

for (i=0;i<kolkon;i++)cout<<fin[i]<<" ";

cout<<endl;

in\_stream.close();

cout<<"End of output file..."<<endl;

}

return;};

//показ текущего состояния

void konavt::show\_sost()

{int i;

cout<<"sostoyanie "<<avtstate<<endl;

cout<<dlina<<endl;

//cout<<"ostatok vxoda "<<vxod<<endl;

cout<<"ostatok vxoda ";

//for (i=0; i<dlina;i++) cout<<vxod[i]<<"\t"; // \*(vxod+i)

//cout<<endl;

for (i=0; i<dlina;i++) cout<<\*(vxod+i)<<"\t";

cout<<endl<<"protokol ";

for (i=0;i<dlina0+1;i++) cout<<protokol[i]<<"\t";

cout<<endl;

};

//переход к следующему состоянию

void konavt::sled()

{int slsost,i,num;

num=-1;

char teksimv;

teksimv=vxod[0];

cout<<"symbol "<<teksimv<<endl;

for (i=0;i<kolsimv;i++) if (teksimv==alfavit[i]) {num=i;break;};

if (num==-1){cout<<"illegal symbol "<<teksimv<<endl;error=true;}

else {slsost=per[avtstate][num];

avtstate=slsost;

protokol[nomstep]=slsost;

nomstep++;

for (i=0;i<dlina;i++)vxod[i]=vxod[i+1];

vxod[dlina-1]=' ';

dlina=dlina-1;

};

return;};

//проверка допустимости входной строки

bool konavt::proverka()

{int i,j;

bool prizn;

for (i=0;i<dlina;i++)

{prizn=false;

for (j=0;j<kolsimv;j++) if (vxod[i]==alfavit[j]) {prizn=true;break;};

if (!prizn) {cout<<"illegal symbol "<<endl;break;};

};

return prizn;

};

//ввод новой входной строки

void konavt::init()

{int i;

cout<<"enter dlina"<<endl;

cin>>dlina;

dlina0=dlina;

vxod=new char[dlina+1];

protokol=new int [dlina+1];

for (i=0;i<dlina+1;i++)protokol[i]=0;

cout<<endl<<"enter vhodnaya stroka"<<endl;

for (i=0;i<dlina;i++)cin>>vxod[i];

//vxod[dlina]='\0';

cout<<endl;

//cout<<"ostatok vxoda "<<vxod<<endl;

cout<<"ostatok vxoda ";

for (i=0; i<dlina;i++) cout<<vxod[i]<<"\t";

cout<<endl;

if (proverka()) {avtstate=nachstate;

protokol[0]=nachstate;

nomstep=1;

error=false;}

else error=true;

return;};

//показ параметров автомата

void konavt::show()

{int i,j;

cout<<"parametry avtomata"<<endl;

cout<<"kolvo sostoianiy "<<kolsost<<endl;

cout<<"kolvo simvolov alfavita "<<kolsimv<<endl;

cout<<"simvoly alfavita"<<endl;

for (j=0;j<kolsimv;j++){

cout<<alfavit[j]<<"\t";

};

cout<<endl<<"matrica perehodov"<<endl;

for (i=0;i<kolsost;i++){

for (j=0;j<kolsimv;j++){

cout<<per[i][j]<<"\t";

};

cout<<endl;

};

cout<<"nachalnoe sostoianie "<<nachstate<<endl;

cout<<"konechnye sostoiania "<<endl;

for (i=0;i<kolkon;i++){

cout<<fin[i]<<"\t";

};cout<<endl;

return;};

//проверка завершающего состояния

bool konavt::konec()

{int i;

bool kon;

kon=false;

i=-1;

for (i=0;i<kolkon;i++) if(avtstate==fin[i]){kon=true;break;};

return kon;

};

//проверка исчерпания входной строки

bool konavt::zaversh()

{bool prizn;

if(dlina==0) prizn=true; else prizn=false;

return prizn;

};

Основная программа main.cpp

# include "konavt.h"

# include <iostream.h>

int main()

{

konavt tavt;

tavt.show();

int povt;

povt=1;

while (povt==1)

{ tavt.init();

tavt.show\_sost();

if (!tavt.error)

{do

{ tavt.sled();

tavt.show\_sost();

} while (!((tavt.error)&&(tavt.zaversh())));

if ((tavt.zaversh()) && (tavt.konec()))

cout<<"\n !!!stroka prinimaetsa"<<endl;else

cout<<"\n !!!stroka ne prinimaetsa"<<endl;}

cout<<"repeat?(1,0)\n";

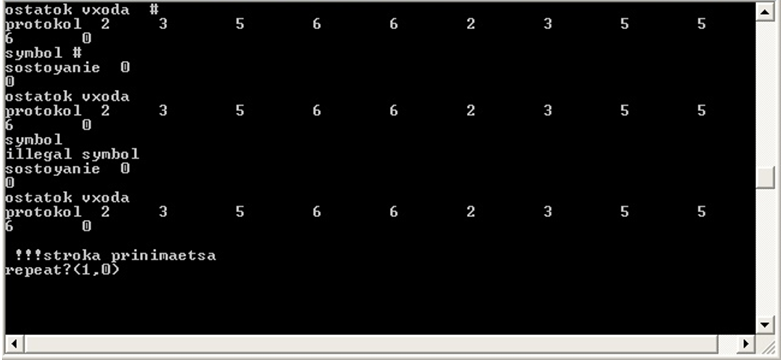
cin>>povt;

cout<<endl;

};

return 0;}

Результат выполнения программы при введении той же строки – q.ad/\_1.a#:



**Конструирование и программное моделирование лексического анализатора**

Цель работы:

Целью работы является получение навыков построения и программного моделирования лексического анализатора (ЛА).

Для выполнения лабораторной работы необходимо:

1) задать синтаксис нескольких лексем средствами описания регулярного языка;

2) построить конечные автоматы для их распознавания;

3) построить сканер, работающий в режиме прямого лексического анализа;

4) написать программу моделирования сканера;

5) программно моделировать построенный сканер на различных примерах.

**Выполнение**

На этапе синтаксического анализа токены иерархически группируются в грамматические фразы, используемые компилятором для синтеза вывода.

Приведённый вариант синтаксического анализатора использует алгоритм Эрли. Т. е. строится список разбора, состоящий из списков ситуаций и затем рекурсивно выписывается правый разбор выражения (входного потока лексем) по грамматике следующего вида

O -> A | O O

A -> M ; | { O } | while V A | for ( j\* = E ; V ; M ) A

A -> if ( V ) A | if ( V ) A else A

M -> j\* x\* E | j\* ++ | j\* -- | ++ j\* | -- j\*

V -> ( E b\* E ) | ( V l\* V ) | E b\* E | V l\* V

E -> T + E | E - T | T

T -> F + T | T / F | F

F -> ( E ) | j\* | c\*

здесь j\* - любое имя переменной, c\* - любая константа, O, A, M, V, E, T, F – нетерминальные символы

#include <fstream>

#include <string>

#include <map> //подключение библиотек

using namespace std;

ifstream cin\_tok("inp.txt"); // файл с входным потоком лексем

ifstream cin\_gr("gram.txt"); // -\\- грамматикой

ifstream cin\_key("key.txt"); // -\\- ключевыми словами

ifstream cin\_res("res.txt"); // -\\- переменными и константами

ofstream cout\_dop("dop.txt"); //файл для списка разбора

ofstream cout\_ans("out.txt"); // -\\- выходной последовательности правил (разбора)

const int MaxRules = 100;

const int MaxLen = 100;

const int MaxNum = 100; //константы

int num\_rules = 0; //количество правил в грамматике

int num\_tok = 0; //количество токенов во входной цепочке

struct pare\_ch //структура пара символов

{

char ch1, ch2;

pare\_ch() //конструктор по умолчанию

{

ch1 = '!';

ch2 = '!';

}

pare\_ch(char c1, char c2) //создание поры из заданных символов

{

ch1 = c1;

ch2 = c2;

}

};

bool operator ==(pare\_ch x, pare\_ch y)

//определение оператора == для переменных типа pare\_ch

{

if (x.ch1 == y.ch1 && (x.ch2 == y.ch2 || x.ch2 == '\*' || y.ch2 == '\*'))

return true;

//равенство пар – попарное равенство 1ых и 2ых символов, при этом второй символ может быть //«\*», которая равна любому символу

return false;

}

bool operator <(pare\_ch x, pare\_ch y)

//определение оператора < для переменных типа pare\_ch

{

if (x.ch1 < y.ch1 || x.ch1 == y.ch1 && x.ch2 < y.ch2)

return true; //лексикографическое сравнение

return false;

}

map <pare\_ch, string> value; //контейнер, выдающий идентификатор по его обозначению

struct rule //структура для правил грамматики

{

char net; //нетерминал

int num; //длина правила

pare\_ch rul[MaxLen]; //само правило в виде массива

} rules[MaxRules]; //массив правил

typedef struct situation //тип «ситуация»

{

pare\_ch net; //нетерминал

int before, after; //количество символов до и после точки соответственно

pare\_ch chein[MaxLen]; //цепочка из терминалов и нетерминалов

int num\_s; //номер списка, где впервые появилось правило

} sit[MaxNum]; //тип – массив ситуаций

struct razbor //структура для списка разбора

{

sit list\_sit; //список ситуаций

int num; //количество ситуаций

} I[MaxNum]; //список разбора

bool operator ==(situation x, situation y)

//определение оператора == для ситуаций

{

bool fl = false;

if (x.net == y.net

&& x.after == y.after

&& x.before == y.before

&& x.num\_s == y.num\_s)

//необходимое условие – равенство нетерминалов, количества символов до и после точки...

{

fl = true;

for (int t = 0; t < y.before + y.after; t++) //проверка совпадения цепочек

{

if (!(x.chein[t] == y.chein[t]))

{

fl = false;

break;

}

}

}

return fl;

}

void Read\_rules() //чтение правил грамматики

{

cin\_gr >> num\_rules; //количество правил

for (int i = 0; i < num\_rules; i++)

{

string s;

cin\_gr >> rules[i].net >> s;

for (int j = 0; j < (int)s.length(); j+=2)

{

rules[i].rul[rules[i].num] = pare\_ch(s[j], s[j + 1]);

rules[i].num++;

}

}

}

void print\_spisok(int i) //вывод i-го списка ситуаций

{

cout\_dop << "I" << i << endl;

for (int w = 0; w < I[i].num; w++) //для всех правил из списка

{

cout\_dop << "[" << I[i].list\_sit[w].net.ch1 << " -> ";

//вывод нетерминала

for (int j = 0; j < I[i].list\_sit[w].before; j++)

//вывод части правила до точки

{

cout\_dop << I[i].list\_sit[w].chein[j].ch1 << I[i].list\_sit[w].chein[j].ch2;

}

cout\_dop << ".";

for (int j = I[i].list\_sit[w].before; j < I[i].list\_sit[w].before + I[i].list\_sit[w].after; j++)

//вывод части правила после точки

{

cout\_dop << I[i].list\_sit[w].chein[j].ch1 << I[i].list\_sit[w].chein[j].ch2;

}

cout\_dop << ", " << I[i].list\_sit[w].num\_s << "] " << "\n";

//вывод соответствующего ситуации номера списка и перевод строки

}

}

void Make\_spisok\_razbora() //создание списков ситуаций (списка разбора)

{

//создание списка I0

for (int i = 0; i < num\_rules; i++)

{

//добавление правил из грамматики

I[0].list\_sit[I[0].num].net = pare\_ch(rules[i].net, '\_');

I[0].list\_sit[I[0].num].before = 0;

I[0].list\_sit[I[0].num].after = rules[i].num;

for (int j = 0; j < rules[i].num; j++)

I[0].list\_sit[I[0].num].chein[j] = rules[i].rul[j];

I[0].list\_sit[I[0].num].num\_s = 0;

I[0].num++;

}

print\_spisok(0); //вывод 0-го списка

//построение остальных списковIi

cin\_tok >> num\_tok; //чтение количества нетерминалов во входной цепочке

for (int i = 1; i <= num\_tok; i++)

{

pare\_ch ch;

string s;

cin\_tok >> s;

ch = pare\_ch(s[0], s[1]);

//добавление в список правил из предыдущего списка, в которых точка стоит перед

//текущим терминалом

for (int j = 0; j < I[i - 1].num; j++)

{

//перебираем все правила предыдущего списка

if (I[i - 1].list\_sit[j].after != 0

&& I[i - 1].list\_sit[j].chein[I[i - 1].list\_sit[j].before] == ch)

{

//проверяем, подходит ли правило и добавляем в новый список

I[i].list\_sit[I[i].num] = I[i - 1].list\_sit[j];

I[i].list\_sit[I[i].num].chein[I[i].list\_sit[I[i].num].before] = ch;

I[i].list\_sit[I[i].num].before++;

I[i].list\_sit[I[i].num].after--;

//переносим точку на символ дальше

I[i].num++;

}

}

bool f = false; //флаг добавления правила

while (!f)

{

//пока в список не перестанут добавляться новые правила

f = true;

//обрабатываем ситуации, в которых точка стоит перед нетерминалом

for (int j = 0; j < I[i].num; j++)

{

if (I[i].list\_sit[j].chein[I[i].list\_sit[j].before].ch2 == '\_')

{

for (int k = 0; k < num\_rules; k++)

{

if (rules[k].net == I[i].list\_sit[j].chein[I[i].list\_sit[j].before].ch1)

{

situation y;

y.net = pare\_ch(rules[k].net, '\_');

y.before = 0;

y.after = rules[k].num;

//проверяем наличие правила, которое собираемся добавить

for (int z = 0; z < rules[k].num; z++)

{

y.chein[z] = rules[k].rul[z];

y.num\_s = i;

}

bool fl = false;

for (int z = 0; z < I[i].num; z++)

if (y == I[i].list\_sit[z])

fl = true;

if (!fl)

{

//добавляем из грамматики правило для данного нетерминала

I[i].list\_sit[I[i].num] = y;

I[i].num++;

f = false;

}

}

}

}

}

//если всё правило стоит перед точкой

for (int j = 0; j < I[i].num; j++)

{

if (I[i].list\_sit[j].after == 0)

{

int num = I[i].list\_sit[j].num\_s;

for (int z = 0; z < I[num].num; z++)

{

if (I[num].list\_sit[z].chein[I[num].list\_sit[z].before] == I[i].list\_sit[j].net)

{

bool fl = false;

situation y;

y = I[num].list\_sit[z];

y.before++;

y.after--;

//проверяем наличие добавляемого правила

for (int z = 0; z < I[i].num; z++)

if (y == I[i].list\_sit[z])

fl = true;

if (!fl)

{

//добавляем правило в список

I[i].list\_sit[I[i].num] = y;

I[i].num++;

f = false;

}

}

}

}

}

}

print\_spisok(i); //выводим список

}

}

void Read\_v\_c\_key()

//чтение ключевых слов, операций, констант, переменных и их обозначений

{

int n;

char ch;

string s;

cin\_key >> n >> ch;

for (int i = 0; i < n; i++)

{

//чтение ключевых слов

cin\_key >> s;

value[pare\_ch(ch, i + '0')] = s;

}

cin\_key >> n;

int m;

for (int i = 0; i < n; i++)

{

//чтение разного вида операций

cin\_key >> m >> ch;

for (int j = 0; j < m; j++)

{

cin\_key >> s;

value[pare\_ch(ch, j + '0')] = s;

}

}

cin\_res >> n;

for (int i = 0; i < n; i++)

{

//чтение имён переменных

cin\_res >> s;

value[pare\_ch('j', i + '0')] = s;

}

cin\_res >> n;

for (int i = 0; i < n; i++)

{

//чтение констант

cin\_res >> s;

value[pare\_ch('c', i + '0')] = s;

}

}

void Print\_Rule(situation sit) //распечатка правила для вывода разбора

{

cout\_ans << sit.net.ch1 << " -> ";

for (int i = 0; i < sit.before + sit.after; i++)

if (sit.chein[i].ch2 == '\_')

{

//вывод нетерминала

cout\_ans << sit.chein[i].ch1 << " ";

}

else

{

//вывод терминала

cout\_ans << value[sit.chein[i]] << " ";

}

cout\_ans << endl;

}

void Rec(situation sit, int num\_sp) //рекурсивная процедура вывода правого разбора

{

Print\_Rule(sit);

int k, l, r;

l = num\_sp;

k = sit.before - 1;

while (k > -1)

{

situation temp = sit;

temp.after += temp.before - k;//a = (b + a)/all/ - (k + 1)/new before/

temp.before = k;

bool f = false;

if (sit.chein[k].ch2 == '\_') //если текущий символ - нетерминал

{

for (int i = 0; i < num\_tok + 1; i++)

{

for (int j = 0; j < I[i].num; j++)

{

if (I[i].list\_sit[j] == temp)

{

for (int z = 0; z < I[l].num; z++)

{

if (I[l].list\_sit[z].net == sit.chein[k]

&& I[l].list\_sit[z].after == 0

&& I[l].list\_sit[z].num\_s == i)

{

r = i;

Rec(I[l].list\_sit[z], l);

//запуск рекурсии от найденной ситуации

f = true;

k--;

l = r;

break;

}

}

}

if (f)

break;

}

if (f)

break;

}

}

else //если текущий символ терминальный

{

k--;

l--;

}

}

}

void Analiz() //проверка на наличие ошибки и поиск 1ой ситуации

{

Read\_v\_c\_key();

//поиск начальной ситуации для рекурсии

bool f = false;

for (int i = 0; i < I[num\_tok].num; i++)

{

if (I[num\_tok].list\_sit[i].after == 0

&& I[num\_tok].list\_sit[i].net.ch1 == rules[0].net

&& I[num\_tok].list\_sit[i].num\_s == 0)

{

f = true;

Rec(I[num\_tok].list\_sit[i], num\_tok);

break;

}

}

if (!f)

{

//если в последнем списке не найдена ситуация нужного вида, цепочка не может быть выведена //по данной грамматике, т. е. в исходном коде присутствует синтаксическая ошибка

cout\_ans << "---------ERROR!!!---------";

}

}

int main()

{

Read\_rules(); //чтение правил грамматики

Make\_spisok\_razbora(); //создание списка разбора

Analiz(); //вывод разбора

return 0;

}

Примеры входных файлов и результатов работы синтаксического анализатора на них.

inp.txt, res.txt и key.txt берутся из лексического анализа

gram.txt

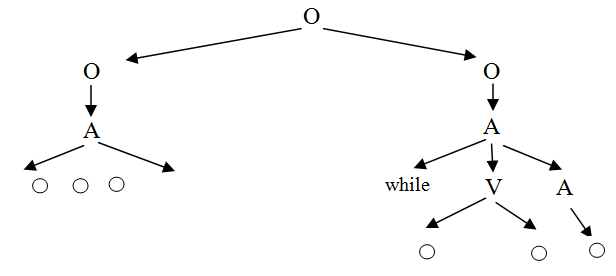
|  |  |
| --- | --- |
| 26  O A\_  O O\_O\_  A M\_r0  A q0O\_q1  A k0V\_A\_  A k3s0j\*x0E\_r0V\_r0M\_s1A\_  A k1s0V\_s1A\_  A k1s0V\_s1A\_k2A\_  M j\*x\*E\_  M j\*p0  M j\*p1  M p0j\*  M p1j\* | V s0E\_b\*E\_s1  V s0V\_l\*V\_s1  V E\_b\*E\_  V V\_l\*V\_  E T\_a0E\_  E E\_a1T\_  E T\_  T F\_a2T\_  T T\_a3F\_  T F\_  F s0E\_s1  F j\*  F c\* |

dop.txt

|  |  |
| --- | --- |
| I0  [O -> .A\_, 0]  [O -> .O\_O\_, 0]  [A -> .M\_r0, 0]  [A -> .q0O\_q1, 0]  [A -> .k0V\_A\_, 0]  [A -> .k3s0j\*x0E\_r0V\_r0M\_s1A\_, 0]  [A -> .k1s0V\_s1A\_, 0]  [A -> .k1s0V\_s1A\_k2A\_, 0]  [M -> .j\*x\*E\_, 0]  [M -> .j\*p0, 0]  [M -> .j\*p1, 0]  [M -> .p0j\*, 0]  [M -> .p1j\*, 0]  [V -> .s0E\_b\*E\_s1, 0]  [V -> .s0V\_l\*V\_s1, 0]  [V -> .E\_b\*E\_, 0]  [V -> .V\_l\*V\_, 0]  [E -> .T\_a0E\_, 0]  [E -> .E\_a1T\_, 0]  [E -> .T\_, 0]  [T -> .F\_a2T\_, 0]  [T -> .T\_a3F\_, 0]  [T -> .F\_, 0]  [F -> .s0E\_s1, 0]  [F -> .j\*, 0]  [F -> .c\*, 0]  I1  [A -> k3.s0j\*x0E\_r0V\_r0M\_s1A\_, 0]  I2  [A -> k3s0.j\*x0E\_r0V\_r0M\_s1A\_, 0]  I3  [A -> k3s0j0.x0E\_r0V\_r0M\_s1A\_, 0]  I4  [A -> k3s0j0x0.E\_r0V\_r0M\_s1A\_, 0]  [E -> .T\_a0E\_, 4]  [E -> .E\_a1T\_, 4]  [E -> .T\_, 4]  [T -> .F\_a2T\_, 4]  [T -> .T\_a3F\_, 4]  [T -> .F\_, 4]  [F -> .s0E\_s1, 4]  [F -> .j\*, 4]  [F -> .c\*, 4]  I5  [F -> c0., 4]  [T -> F\_.a2T\_, 4]  [T -> F\_., 4]  [E -> T\_.a0E\_, 4]  [E -> T\_., 4]  [T -> T\_.a3F\_, 4]  [A -> k3s0j0x0E\_.r0V\_r0M\_s1A\_, 0]  [E -> E\_.a1T\_, 4]  I6  [A -> k3s0j0x0E\_r0.V\_r0M\_s1A\_, 0]  [V -> .s0E\_b\*E\_s1, 6]  [V -> .s0V\_l\*V\_s1, 6]  [V -> .E\_b\*E\_, 6]  [V -> .V\_l\*V\_, 6]  [E -> .T\_a0E\_, 6]  [E -> .E\_a1T\_, 6]  [E -> .T\_, 6]  [T -> .F\_a2T\_, 6]  [T -> .T\_a3F\_, 6]  [T -> .F\_, 6]  [F -> .s0E\_s1, 6]  [F -> .j\*, 6]  [F -> .c\*, 6]  I7  [F -> j0., 6]  [T -> F\_.a2T\_, 6]  [T -> F\_., 6]  [E -> T\_.a0E\_, 6]  [E -> T\_., 6]  [T -> T\_.a3F\_, 6]  [V -> E\_.b\*E\_, 6]  [E -> E\_.a1T\_, 6]  I8  [V -> E\_b1.E\_, 6]  [E -> .T\_a0E\_, 8]  [E -> .E\_a1T\_, 8]  [E -> .T\_, 8]  [T -> .F\_a2T\_, 8]  [T -> .T\_a3F\_, 8]  [T -> .F\_, 8]  [F -> .s0E\_s1, 8]  [F -> .j\*, 8]  [F -> .c\*, 8]  I9  [F -> j1., 8]  [T -> F\_.a2T\_, 8]  [T -> F\_., 8]  [E -> T\_.a0E\_, 8]  [E -> T\_., 8]  [T -> T\_.a3F\_, 8]  [V -> E\_b1E\_., 6]  [E -> E\_.a1T\_, 8]  [A -> k3s0j0x0E\_r0V\_.r0M\_s1A\_, 0]  [V -> V\_.l\*V\_, 6]  I10  [A -> k3s0j0x0E\_r0V\_r0.M\_s1A\_, 0]  [M -> .j\*x\*E\_, 10]  [M -> .j\*p0, 10]  [M -> .j\*p1, 10]  [M -> .p0j\*, 10]  [M -> .p1j\*, 10]  I11  [M -> j0.x\*E\_, 10]  [M -> j0.p0, 10]  [M -> j0.p1, 10]  I12  [M -> j0p0., 10]  [A -> k3s0j0x0E\_r0V\_r0M\_.s1A\_, 0]  I13  [A -> k3s0j0x0E\_r0V\_r0M\_s1.A\_, 0]  [A -> .M\_r0, 13]  [A -> .q0O\_q1, 13]  [A -> .k0V\_A\_, 13]  [A -> .k3s0j\*x0E\_r0V\_r0M\_s1A\_, 13]  [A -> .k1s0V\_s1A\_, 13]  [A -> .k1s0V\_s1A\_k2A\_, 13]  [M -> .j\*x\*E\_, 13]  [M -> .j\*p0, 13]  [M -> .j\*p1, 13]  [M -> .p0j\*, 13]  [M -> .p1j\*, 13]  I14  [A -> q0.O\_q1, 13]  [O -> .A\_, 14]  [O -> .O\_O\_, 14]  [A -> .M\_r0, 14]  [A -> .q0O\_q1, 14]  [A -> .k0V\_A\_, 14]  [A -> .k3s0j\*x0E\_r0V\_r0M\_s1A\_, 14]  [A -> .k1s0V\_s1A\_, 14]  [A -> .k1s0V\_s1A\_k2A\_, 14]  [M -> .j\*x\*E\_, 14]  [M -> .j\*p0, 14]  [M -> .j\*p1, 14]  [M -> .p0j\*, 14]  [M -> .p1j\*, 14]  I15  [A -> k1.s0V\_s1A\_, 14]  [A -> k1.s0V\_s1A\_k2A\_, 14]  I16  [A -> k1s0.V\_s1A\_, 14]  [A -> k1s0.V\_s1A\_k2A\_, 14]  [V -> .s0E\_b\*E\_s1, 16]  [V -> .s0V\_l\*V\_s1, 16]  [V -> .E\_b\*E\_, 16]  [V -> .V\_l\*V\_, 16]  [E -> .T\_a0E\_, 16]  [E -> .E\_a1T\_, 16]  [E -> .T\_, 16]  [T -> .F\_a2T\_, 16]  [T -> .T\_a3F\_, 16]  [T -> .F\_, 16]  [F -> .s0E\_s1, 16]  [F -> .j\*, 16]  [F -> .c\*, 16]  I17  [F -> j2., 16]  [T -> F\_.a2T\_, 16]  [T -> F\_., 16]  [E -> T\_.a0E\_, 16]  [E -> T\_., 16]  [T -> T\_.a3F\_, 16]  [V -> E\_.b\*E\_, 16]  [E -> E\_.a1T\_, 16]  I18  [V -> E\_b2.E\_, 16]  [E -> .T\_a0E\_, 18]  [E -> .E\_a1T\_, 18]  [E -> .T\_, 18]  [T -> .F\_a2T\_, 18]  [T -> .T\_a3F\_, 18]  [T -> .F\_, 18]  [F -> .s0E\_s1, 18]  [F -> .j\*, 18]  [F -> .c\*, 18]  I19  [F -> j3., 18]  [T -> F\_.a2T\_, 18]  [T -> F\_., 18]  [E -> T\_.a0E\_, 18]  [E -> T\_., 18]  [T -> T\_.a3F\_, 18]  [V -> E\_b2E\_., 16]  [E -> E\_.a1T\_, 18]  [A -> k1s0V\_.s1A\_, 14]  [A -> k1s0V\_.s1A\_k2A\_, 14]  [V -> V\_.l\*V\_, 16]  I20  [A -> k1s0V\_s1.A\_, 14]  [A -> k1s0V\_s1.A\_k2A\_, 14]  [A -> .M\_r0, 20]  [A -> .q0O\_q1, 20]  [A -> .k0V\_A\_, 20]  [A -> .k3s0j\*x0E\_r0V\_r0M\_s1A\_, 20]  [A -> .k1s0V\_s1A\_, 20]  [A -> .k1s0V\_s1A\_k2A\_, 20]  [M -> .j\*x\*E\_, 20]  [M -> .j\*p0, 20]  [M -> .j\*p1, 20]  [M -> .p0j\*, 20]  [M -> .p1j\*, 20] I21  [A -> q0.O\_q1, 20]  [O -> .A\_, 21]  [O -> .O\_O\_, 21]  [A -> .M\_r0, 21]  [A -> .q0O\_q1, 21]  [A -> .k0V\_A\_, 21]  [A -> .k3s0j\*x0E\_r0V\_r0M\_s1A\_, 21]  [A -> .k1s0V\_s1A\_, 21]  [A -> .k1s0V\_s1A\_k2A\_, 21] | [M -> .j\*x\*E\_, 21]  [M -> .j\*p0, 21]  [M -> .j\*p1, 21]  [M -> .p0j\*, 21]  [M -> .p1j\*, 21]  I22  [M -> j3.x\*E\_, 21]  [M -> j3.p0, 21]  [M -> j3.p1, 21]  I23  [M -> j3x0.E\_, 21]  [E -> .T\_a0E\_, 23]  [E -> .E\_a1T\_, 23]  [E -> .T\_, 23]  [T -> .F\_a2T\_, 23]  [T -> .T\_a3F\_, 23]  [T -> .F\_, 23]  [F -> .s0E\_s1, 23]  [F -> .j\*, 23]  [F -> .c\*, 23]  I24  [F -> j2., 23]  [T -> F\_.a2T\_, 23]  [T -> F\_., 23]  [E -> T\_.a0E\_, 23]  [E -> T\_., 23]  [T -> T\_.a3F\_, 23]  [M -> j3x0E\_., 21]  [E -> E\_.a1T\_, 23]  [A -> M\_.r0, 21]  I25  [A -> M\_r0., 21]  [O -> A\_., 21]  [A -> q0O\_.q1, 20]  [O -> O\_.O\_, 21]  [O -> .A\_, 25]  [O -> .O\_O\_, 25]  [A -> .M\_r0, 25]  [A -> .q0O\_q1, 25]  [A -> .k0V\_A\_, 25]  [A -> .k3s0j\*x0E\_r0V\_r0M\_s1A\_, 25]  [A -> .k1s0V\_s1A\_, 25]  [A -> .k1s0V\_s1A\_k2A\_, 25]  [M -> .j\*x\*E\_, 25]  [M -> .j\*p0, 25]  [M -> .j\*p1, 25]  [M -> .p0j\*, 25]  [M -> .p1j\*, 25]  I26  [A -> q0O\_q1., 20]  [A -> k1s0V\_s1A\_., 14]  [A -> k1s0V\_s1A\_.k2A\_, 14]  [O -> A\_., 14]  [A -> q0O\_.q1, 13]  [O -> O\_.O\_, 14]  [O -> .A\_, 26]  [O -> .O\_O\_, 26]  [A -> .M\_r0, 26]  [A -> .q0O\_q1, 26]  [A -> .k0V\_A\_, 26]  [A -> .k3s0j\*x0E\_r0V\_r0M\_s1A\_, 26]  [A -> .k1s0V\_s1A\_, 26]  [A -> .k1s0V\_s1A\_k2A\_, 26]  [M -> .j\*x\*E\_, 26]  [M -> .j\*p0, 26]  [M -> .j\*p1, 26]  [M -> .p0j\*, 26]  [M -> .p1j\*, 26]  I27  [A -> q0O\_q1., 13]  [A -> k3s0j0x0E\_r0V\_r0M\_s1A\_., 0]  [O -> A\_., 0]  [O -> O\_.O\_, 0]  [O -> .A\_, 27]  [O -> .O\_O\_, 27]  [A -> .M\_r0, 27]  [A -> .q0O\_q1, 27]  [A -> .k0V\_A\_, 27]  [A -> .k3s0j\*x0E\_r0V\_r0M\_s1A\_, 27]  [A -> .k1s0V\_s1A\_, 27]  [A -> .k1s0V\_s1A\_k2A\_, 27]  [M -> .j\*x\*E\_, 27]  [M -> .j\*p0, 27]  [M -> .j\*p1, 27]  [M -> .p0j\*, 27]  [M -> .p1j\*, 27]  I28  [A -> k0.V\_A\_, 27]  [V -> .s0E\_b\*E\_s1, 28]  [V -> .s0V\_l\*V\_s1, 28]  [V -> .E\_b\*E\_, 28]  [V -> .V\_l\*V\_, 28]  [E -> .T\_a0E\_, 28]  [E -> .E\_a1T\_, 28]  [E -> .T\_, 28]  [T -> .F\_a2T\_, 28]  [T -> .T\_a3F\_, 28]  [T -> .F\_, 28]  [F -> .s0E\_s1, 28]  [F -> .j\*, 28]  [F -> .c\*, 28]  I29  [V -> s0.E\_b\*E\_s1, 28]  [V -> s0.V\_l\*V\_s1, 28]  [F -> s0.E\_s1, 28]  [E -> .T\_a0E\_, 29]  [E -> .E\_a1T\_, 29]  [E -> .T\_, 29]  [V -> .s0E\_b\*E\_s1, 29]  [V -> .s0V\_l\*V\_s1, 29]  [V -> .E\_b\*E\_, 29]  [V -> .V\_l\*V\_, 29]  [T -> .F\_a2T\_, 29]  [T -> .T\_a3F\_, 29]  [T -> .F\_, 29]  [F -> .s0E\_s1, 29]  [F -> .j\*, 29]  [F -> .c\*, 29]  I30  [F -> j0., 29]  [T -> F\_.a2T\_, 29]  [T -> F\_., 29]  [E -> T\_.a0E\_, 29]  [E -> T\_., 29]  [T -> T\_.a3F\_, 29]  [V -> s0E\_.b\*E\_s1, 28]  [F -> s0E\_.s1, 28]  [E -> E\_.a1T\_, 29]  [V -> E\_.b\*E\_, 29]  I31  [V -> s0E\_b1.E\_s1, 28]  [V -> E\_b1.E\_, 29]  [E -> .T\_a0E\_, 31]  [E -> .E\_a1T\_, 31]  [E -> .T\_, 31]  [T -> .F\_a2T\_, 31]  [T -> .T\_a3F\_, 31]  [T -> .F\_, 31]  [F -> .s0E\_s1, 31]  [F -> .j\*, 31]  [F -> .c\*, 31]  I32  [F -> j4., 31]  [T -> F\_.a2T\_, 31]  [T -> F\_., 31]  [E -> T\_.a0E\_, 31]  [E -> T\_., 31]  [T -> T\_.a3F\_, 31]  [V -> s0E\_b1E\_.s1, 28]  [V -> E\_b1E\_., 29]  [E -> E\_.a1T\_, 31]  [V -> s0V\_.l\*V\_s1, 28]  [V -> V\_.l\*V\_, 29]  I33  [V -> s0E\_b1E\_s1., 28]  [A -> k0V\_.A\_, 27]  [V -> V\_.l\*V\_, 28]  [A -> .M\_r0, 33]  [A -> .q0O\_q1, 33]  [A -> .k0V\_A\_, 33]  [A -> .k3s0j\*x0E\_r0V\_r0M\_s1A\_, 33]  [A -> .k1s0V\_s1A\_, 33]  [A -> .k1s0V\_s1A\_k2A\_, 33]  [M -> .j\*x\*E\_, 33]  [M -> .j\*p0, 33]  [M -> .j\*p1, 33]  [M -> .p0j\*, 33]  [M -> .p1j\*, 33]  I34  [M -> j2.x\*E\_, 33]  [M -> j2.p0, 33]  [M -> j2.p1, 33]  I35  [M -> j2x4.E\_, 33]  [E -> .T\_a0E\_, 35]  [E -> .E\_a1T\_, 35]  [E -> .T\_, 35]  [T -> .F\_a2T\_, 35]  [T -> .T\_a3F\_, 35]  [T -> .F\_, 35]  [F -> .s0E\_s1, 35]  [F -> .j\*, 35]  [F -> .c\*, 35]  I36  [F -> c1., 35]  [T -> F\_.a2T\_, 35]  [T -> F\_., 35]  [E -> T\_.a0E\_, 35]  [E -> T\_., 35]  [T -> T\_.a3F\_, 35]  [M -> j2x4E\_., 33]  [E -> E\_.a1T\_, 35]  [A -> M\_.r0, 33]  I37  [A -> M\_r0., 33]  [A -> k0V\_A\_., 27]  [O -> A\_., 27]  [O -> O\_O\_., 0]  [O -> O\_.O\_, 27]  [O -> O\_.O\_, 0]  [O -> .A\_, 37]  [O -> .O\_O\_, 37]  [A -> .M\_r0, 37]  [A -> .q0O\_q1, 37]  [A -> .k0V\_A\_, 37]  [A -> .k3s0j\*x0E\_r0V\_r0M\_s1A\_, 37]  [A -> .k1s0V\_s1A\_, 37]  [A -> .k1s0V\_s1A\_k2A\_, 37]  [M -> .j\*x\*E\_, 37]  [M -> .j\*p0, 37]  [M -> .j\*p1, 37]  [M -> .p0j\*, 37]  [M -> .p1j\*, 37] |

Содержимое файла результатов работы: out.txt

|  |  |
| --- | --- |
| O -> O O  O -> A  A -> while V A  A -> M ;  M -> a /= E  E -> T  T -> F  F -> 20  V -> ( E < E )  E -> T  T -> F  F -> d  E -> T  T -> F  F -> i  O -> A  A -> for ( i = E ; V ; M ) A  A -> { O }  O -> A  A -> if ( V ) A  A -> { O }  O -> A  A -> M ; | M -> b = E  E -> T  T -> F  F -> a  V -> E == E  E -> T  T -> F  F -> b  E -> T  T -> F  F -> a  M -> i ++  V -> E < E  E -> T  T -> F  F -> n  E -> T  T -> F  F -> i  E -> T  T -> F  F -> 10 |

****

*СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ*

1. Горбатов В.А., Горбатов А.В., Горбатова М.В. Теория автоматов: учебник для студентов втузов. – М.: АСТ: Астрель. – 559 с.
2. Захаров Н.Г., Рогов В.Н. Синтез цифровых автоматов: учебное пособие – Ульяновск: УлГТУ, 2003. – 136 с.
3. Каган Б.М. Электронные вычислительные машины и системы. – М.: Энергоавтомиздат, 1988. – 551 с.
4. Карпов Ю.Г. Теория автоматов: учебник для вузов. - 1-е изд. – СПб.: Питер, 2003. – 208 с.
5. Киносита К., Асада К., Карашу О. Логическое проектирование СБИС. – М.: Мир, 1988. – 309 с.
6. Пухальский Г.И., Новоселова Т.Я. Цифровые устройства. Учебное пособие для вузов. – СПб.: Политехника издательство, 1996. – 880 с.
7. Савельев А.Я. Прикладная теория цифровых автоматов. Учебник для вузов. - М.: Высшая школа, 1987. – 272 с.
8. Савельев Н.В., Коняхин В.В. Функционально-логическое проектирование БИС. – М.: Высшая школа, 1990. – 156 с.