Московский авиационный институт

(национальный исследовательский университет)

Факультет «Прикладная математика и физика»

**Лабораторные работы по курсу**

**«Системное программное обеспечение»**

1. Спроектировать грамматику по заданному языку L

2. Спроектировать конечный автомат, составить диаграмму переходов КА и реализовать

3. Определить свойства КА. Построить НДКА. Реализовать преобразование НДКА в ДКА.

4. Устранить из КС-грамматики бесполезные символы и ε–правила

5 Устранить из KС-грамматики цепные правила и устранить левую рекурсию

6 Определить форму КС-грамматики и сделать ее приведение

7. Спроектировать МП автомат для приведенной КС-грамматики

8. Реализовать МП автомат для приведенной КС-грамматики

9. Для LL(1) анализатора построить управляющую таблицу M

10. Аналитически написать такты работы LL(1) анализатора для выведенной цепочки.

11. Реализовать управляющую таблицу M для LL(1) анализатора.

12. Построить замыкание множества ситуаций для пополненной LR(1) грамматики.

13. Определить функцию перехода g(х)

14. Построить каноническую форму множества ситуаций.

15. Построить управляющую таблицу для функции перехода g(х) и действий f(u).

16. Реализовать LR(1)-анализатор по управляющей таблице (g,f) для LR(1) грамматики.

*Студент:* Тимофеев А.В.

*Группа:*  08-207Б-19

*Руководитель:* Семёнов А. С.

*Оценка:*

*Дата:*

**Москва 2021**

**Вариает 24.**

**Лабораторная № 1.**

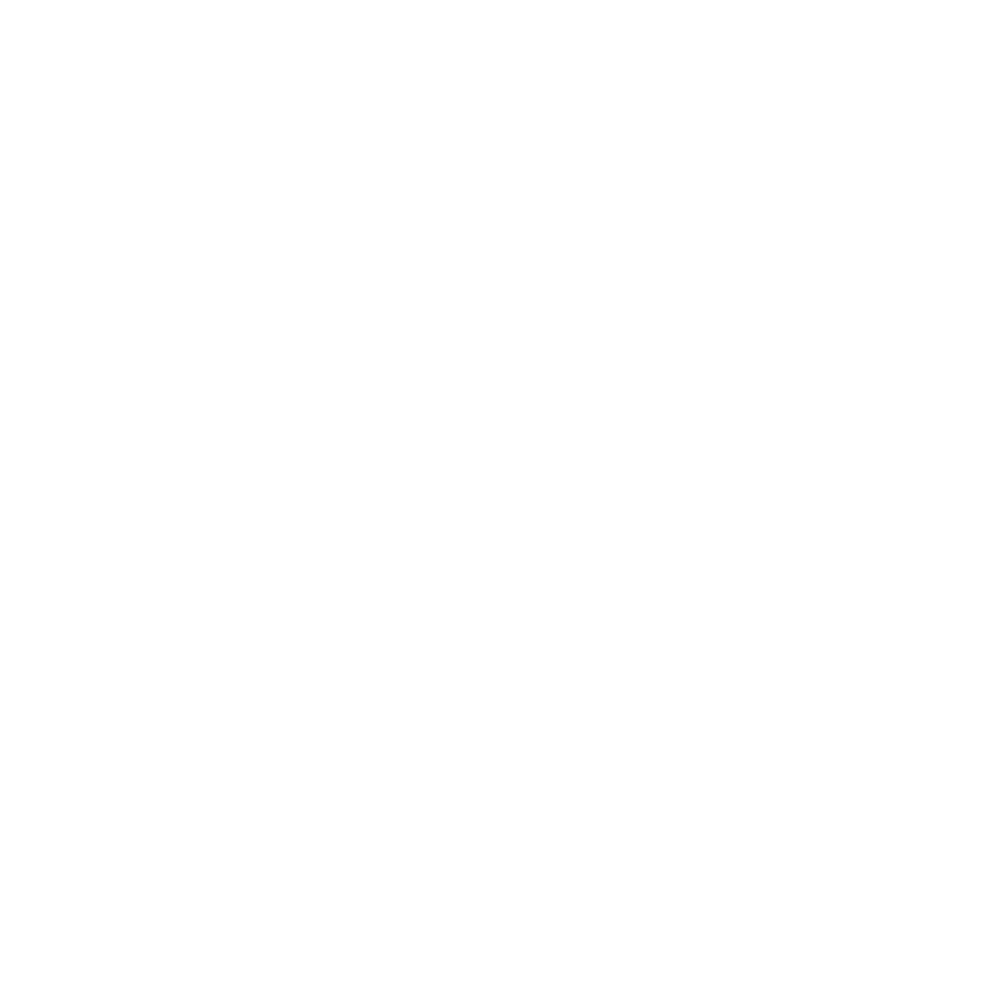
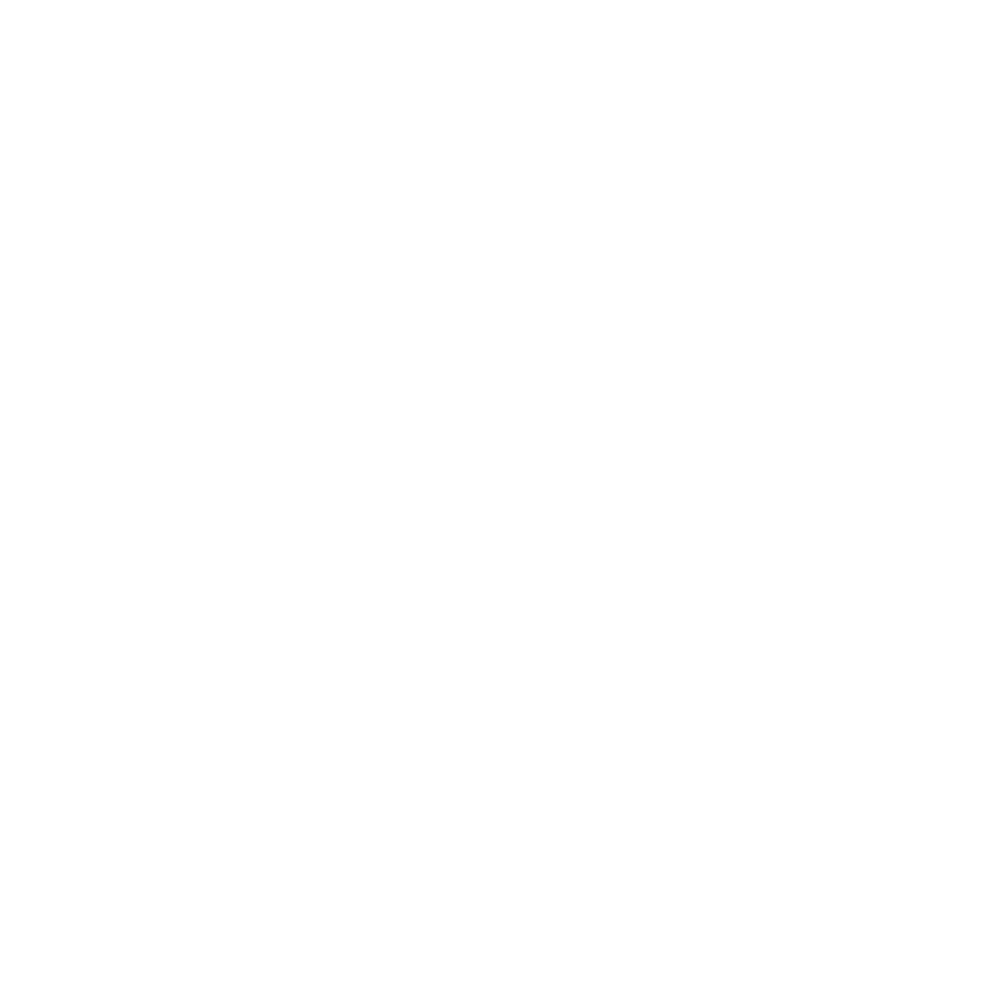
**Формулировка задачи:**

**Спроектировать грамматику по заданному языку L**:

1.1. Задан бесконечный язык L.

1.2. Преобразовать бесконечный язык L в язык L1 представляющий собой конечное множество цепочек, цепочки символов которого являются подмножество цепочек символов бесконечного языка. Ввести ограничения для языка, если они необходимы.

1.3. Сгенерировать цепочки символов по языку L1. Определить свойства языка L1.

24. L={ω1ω20|ω1{1,0}+, ω2{1,0}+}

(1) L = { 110, 100, 000, 1000, 11100 ... } ⊂ L1 = { 110, 100, 000, 1000, 10100 }

# L = { 110, 100, 000, 1000, 11100 ... } ⊂ L1 = { 110, 100, 000, 1000, 10100 }

(2) L1 = L(G) = (язык, порожденный грамматикой ~ заданному)

G = ({1,0},{S0,A,B},P,S0)

p1: S0 → A

p2:A → 0B

p3:A → 1C

p4:C → A

p5:B → A

p6:A → 0

S0 =>1 A =>3 1C =>4 1A =>3 11C =>4 11A =>6 110

S0 =>1 A =>3 1C =>4 1A =>2 10B =>5 10A =>3 101C =>4 101A =>2 1010B =>5 1010A =>6 10100

S0 =>1 A =>3 1C =>4 1A =>2 10B =>5 10A =>6 100

S0 =>1 A =>3 1C =>4 1A =>2 10B =>5 10A =>2 100B =>5 100A =>6 1000

S0 =>1 A =>2 0B =>5 0A =>2 00B =>5 00A =>6  000

**Лабораторная № 2.**

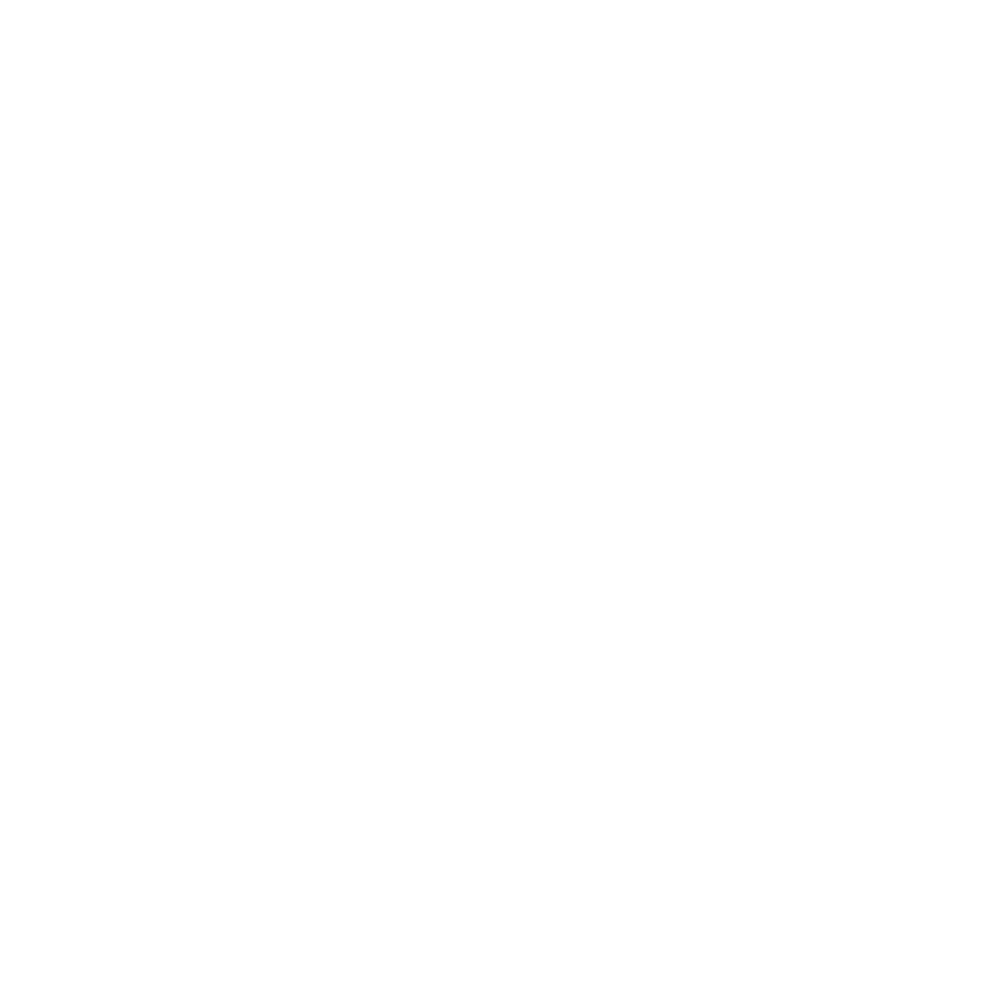
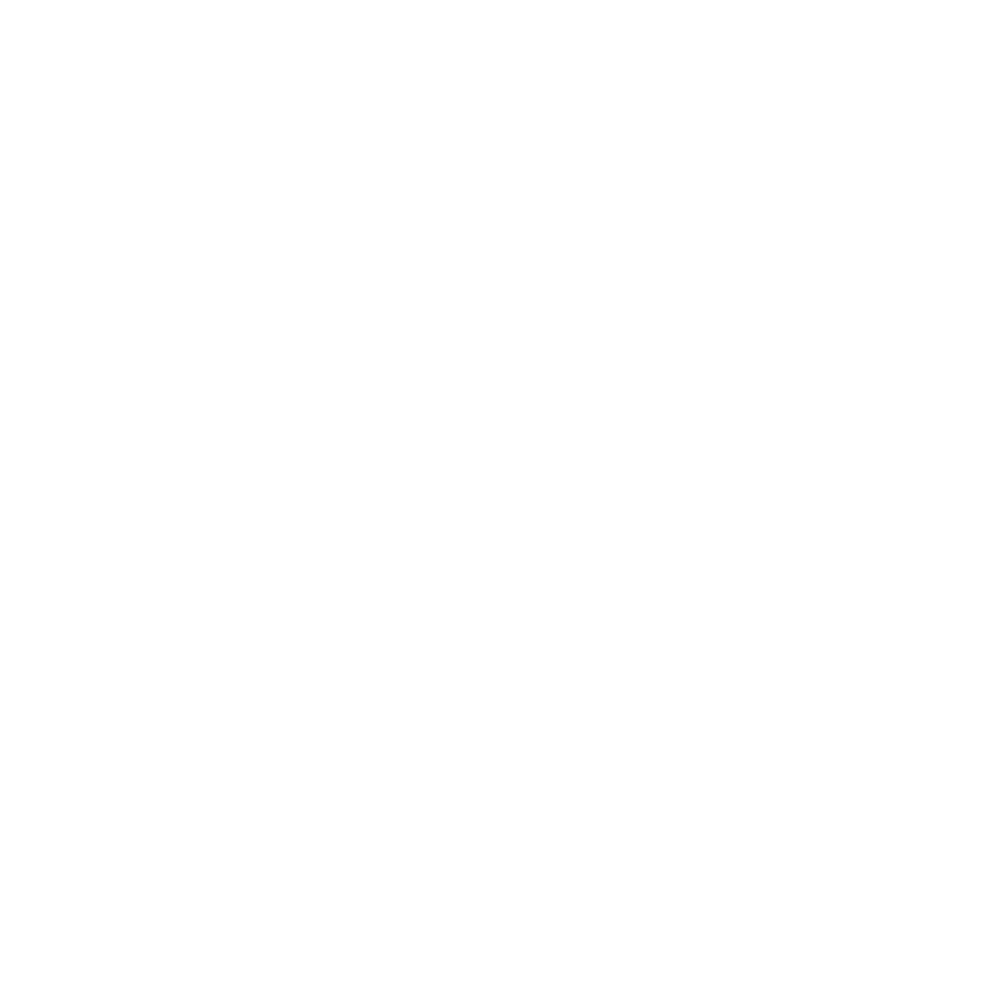
**Формулировка задачи:**

**2.Спроектировать конечный автомат, составить диаграмму переходов КА и реализовать**

2.1.. Спроектировать грамматику для языка L1.

2.2. Построить диаграмму переходов и таблицу переходов по грамматике.

w = 10100  
x = 1  
y = 010  
z = 0  
p = 5

(1) Для всех регулярных языков ∀ L ⊆ E\* (regular(L={ω1ω20|ω1{1,0}+, ω2{1,0}+}) =>   
(2) существует целое ( ∃ p ≥ 1 такое что (p = 5) |  
(3) для всех (∀ 10100 ∈ L (( |10100| ≥ 5) =>   
(4) существует (∃(x = 1; y = 010; z = 0) ∈ E\* такое что (10100 = xyz =>   
1. (| y | = |010| ≥ 1, цикл y должен быть накачан хотя бы длиной 1 и   
2. |xy | = |1010| ≤ p, цикл должен быть в пределах первых p символов и (получается |4| ≤ 5)  
3. для всех i ≥ 0, ( xy i z = 10100 ∈ L)))))))), на x и z ограничений не накладывается.

Конечный автомат:

КА = (Q,∑,δ,q0,F), где Q - конечное мн-во, ∑ - конечный алфавит входящих символов, δ - мн-во ф-ий переходов , задаваемых отображением δ:Q x ∑ ->Q  
(F - конечное мн-во)

Построение КА. Воспользуемся утверждением 1, тогда

P:G = ({1,0},{S0,A,B},P,S0)

S0 => A

A => 0B

A => 1C

C => A

B => A

A => 0

КА = ({0,1},{S0,A,B,qf},δ,S0,qf)

1. δ(S0,\_) = {A}
2. δ(A,0) = {B}
3. δ(A,1) = {C}
4. δ(A,0) = {qf}
5. δ(B,\_) = {A}
6. δ(C,\_) = {A}

Пример конфигурации КА:

(S0,110) |-1 (A,110) |-3 (C,10) |-6 (A,10) |-3 (B,0) |-2 (A,\_) |-4 (qf,ε)

(S0,100) |-1 (A,100) |-3 (C,00) |-6 [ (A,00) |-2 (B,0)|-5 ] \*2 |-4 (qf,ε)

(S0,000) |-1 [ (A,000) |-2 (B,00)|-5 ] \*3 |-4 (qf,ε)

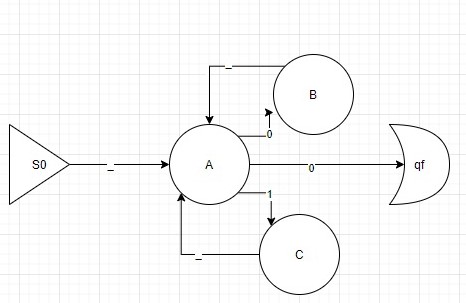
(S0,1000) |-1  (A,1000) |-3 (C,000) |-6 [ (A,000) |-2 (B,00)|-5 ] \*3 |-4 (qf,ε)

Таким образом, конечный автомат имеет вид:

KA = ({S0,A,B,qf},{0,1}, δ,S0,{qf}).

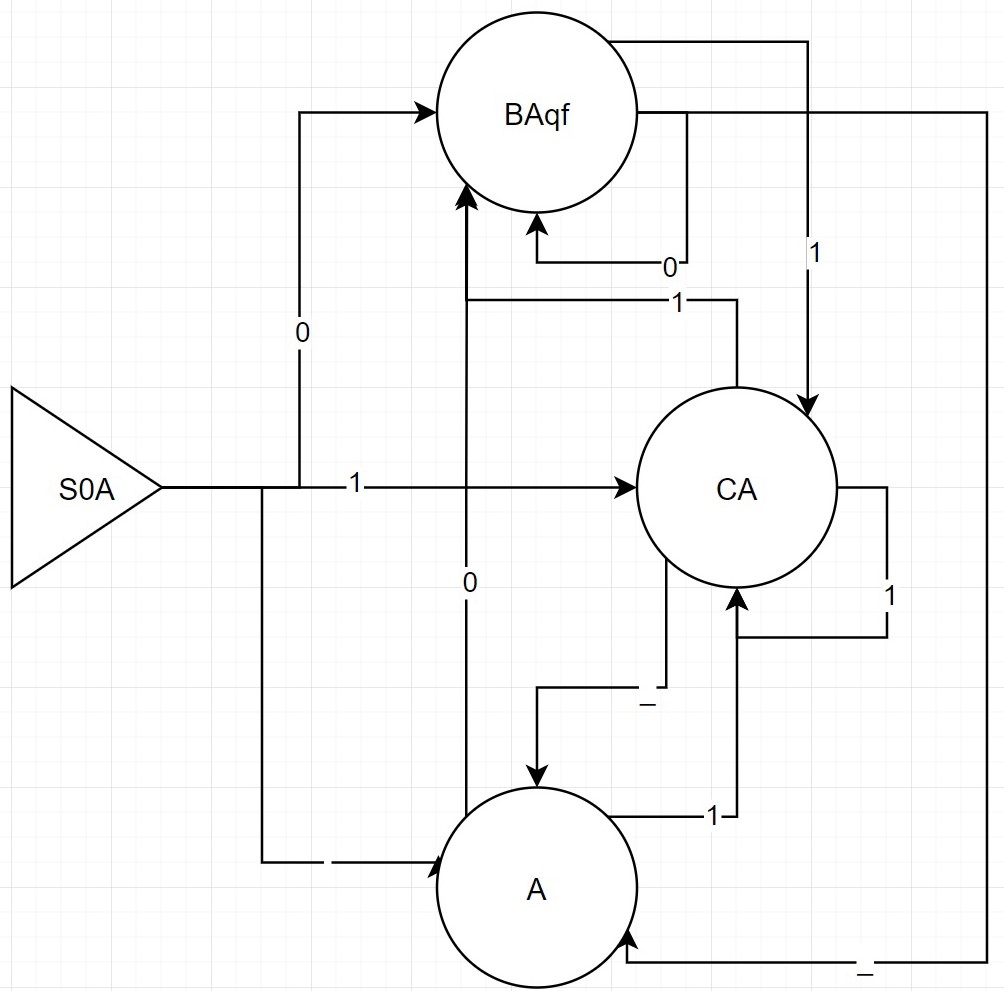
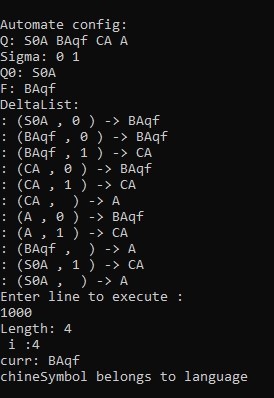
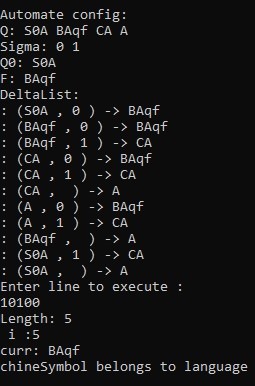
КА является недетерминированным, то есть существуют конфигурации, для которых существует более одного следующего шага.

**Диаграмма переходов конечного автомата**



**Лабораторная № 3.**

**Протокол:**



**Листинг:**

case "2":

myAutomate ndka = new myAutomate(new ArrayList() { "S0", "A", "B", "C", "qf" },

new ArrayList() { "0", "1", "" },

new ArrayList() { "qf" },

"S0");

ndka.AddRule("S0", "", "A");

ndka.AddRule("A", "0", "B");

ndka.AddRule("A", "1", "C");

ndka.AddRule("B", "", "A");

ndka.AddRule("C", "", "A");

ndka.AddRule("A", "", "qf");

myAutomate dka = new myAutomate();

dka.BuildDeltaDKAutomate(ndka);

dka.DebugAuto();

Console.WriteLine("Enter line to execute :");

dka.Execute(Console.ReadLine());

break;

**4. Устранить из КС-грамматики бесполезные символы и ε–правила**

**Задание:**

4.1 Задана КС-грамматика:

P={S → AB, S → C, D → SC, D → d, A→ G, A→CB, G → F, A→ Ab, A→ BB, B → b, C→ a D → ε, C → ε.}

Возможные строки: S => AB => BBB => bbb   
 S => C => B => b

S => AB => CBB => abb

4.2. Устранить бесполезные символы, ε-правила.

**Алгоритм удаление бесполезных символов.**

Вначале исключить непроизводящие нетерминалы, а затем недостижимые символы.

**Алгоритм удаление непроизводящие символов**:

Вход: КС G = (T, V, P, S)

Выход: Vp = {A | A + α, A ∈ V , α ∈ T + }

Алгоритм устранение непроизводящих символов.

Определение множества производящих нетерминальных символов Vp.

Шаг 0. V0 = Пуст. множество.

Шаг 1. V1 = {B,C,D}  
Шаг 2. V2 = {S,A,B,C,D}  
Шаг 3. V3 = {S,A,B,C,D}  
Шаг 4. цикл завершается.

Символ G и F не являются производящими. Удаляем все правила с ними.

G = (V={a,b,d},Vp={S,A,B,C,D},P,S), где P={S → AB, S → C, D → SC, D → d, A→CB, A→ Ab, A→ BB, B → b, C→ a D → ε, C → ε}

S => AB => CBB => abb

**Алгоритм удаление недостижимых символов**

Построить множество VTir - достижимых терминалов и не терминалов

A🠖α. VTr = {x | A 🠖 α, x ∈ α, α ⊆(V ∪ T)\*}

В цикле проверяем все символы каждой цепочки на достижимость и добавляем их в множество VTr. Все символы, которые не попали в это множество, являются недостижимыми.

Недостижимые D и d, устраняем все правила с ними.

G = (V={a,b,d},Vp={S,A,B,C},P,S), где P={S → AB, S → C, A→CB, A→ Ab, A→ BB, B → b, C→ a C → ε}

S => AB => CBB => abb

**Алгоритм удаление эпсилон-правил.**

Грамматика P={S → AB, S → C, A→CB, A→ Ab, A→ BB, B → b, C→ a, C→ B, C → ε}

**Алгоритм построения множества укорачивающих нетерминалов:**

Вход: КС грамматика G = (T,V,P,S).

Выход: Vε = {A | A →+ ε, A∈V}

Шаг 1. Добавляем в множество Viε правила вида A → ε.

Шаг 2. Добавляем в множество Viε правила вида A → B1, B2 ... Bn, где каждый Bi укорачивающийся нетерминал.

Множество укорачивающих терминалов Vε = {C}

1. Положим P’ = пуст. мн-во.
2. Рассмотрим правило:   
   C→ Ca для него в множество P’ добавим правила C→ Ca и C→ a  
   Тогда P’={S → AB, S → C, A→CB, A→ Ab, A→ BB, B → b, C→ a, C→ B}.

4. S принадлежит Vε, тогда положим P’V’ = V

5. G’ = ({a,b},{S,A,B,C},{S → AB, S → C, A→CB, A→ Ab, A→ BB, B → b, C→ a},S).

S => AB => CBB => abb

**5. Устранить из KС-грамматики цепные правила и устранить левую рекурсию**

5.1. **Устранение левой рекурсии** из KС – грамматики G = ({a,b}, {S,A,B,C}, P, S), где P = {S → AB, S → C, A→CB, A→ Ab, A→ BB, B → b, C→ a}.

1. Нумеруем S = A1, A = A2, B = A3, C = A4,

2. Для i = 1 правила вида Ai →Ajα отсутствует.

3. Для i = 2 правила вида Ai →Ajα существует

Заменим A→ BB, A→ Ab, A→ СВ, правилами A→ BB, A→ СВ, A→ bA’, A’→ b, A’→ bA’, где α = b β = BB, CB

4. Для i = 3 правила вида Ai →Ajα отсутствует

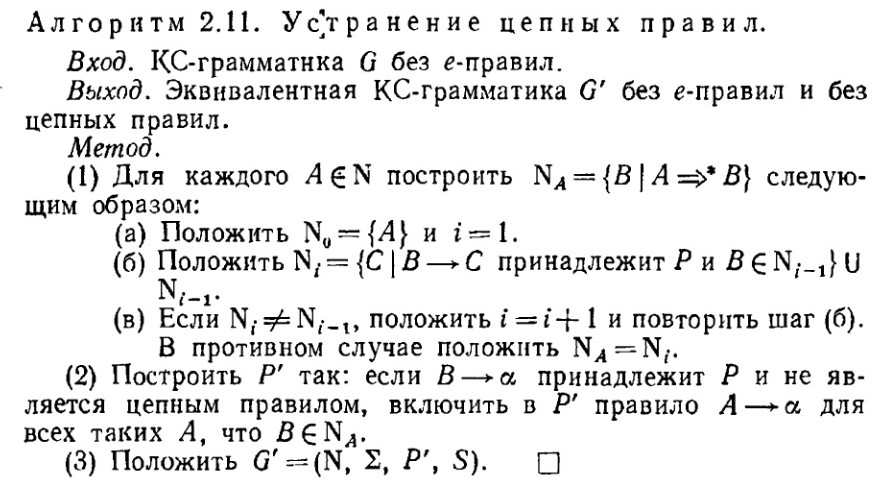
5. Для i = 4 правила вида Ai →Ajα отсутствует

6. Получаем: G’ ={{a,b},{S,A,B,C,S’,A’,B’,C’},P’,S}, где P’ = {S → AB, S → C, A→CB, A→ bA’, A’→ b, A’→ bA’, A→ BB, B → b, C→ a}.

S => AB => CBB => abb

5.2. **Устранение цепных правил** из KС – грамматики   
G = {{a,b},{S,A,B,C,S’,A’,B’,C’},P,S}, где P = {S → AB, S → C, A→CB, A→ bA’, A’→ b, A’→ bA’, A→ BB, B → b, C→ a}.

S → C цепное.



Правило вида A→B, где A и B — нетерминалы называется цепным.

1. Составим множества NS’ = {S, C}, NC = {C}

2. Получим множество P’ = {S → AB, A→CB, A→ bA’, A’→ b, A’→ bA’, A→ BB, B → b, C→ a}

3. Получаем: G’ = (T, V, P’,S)

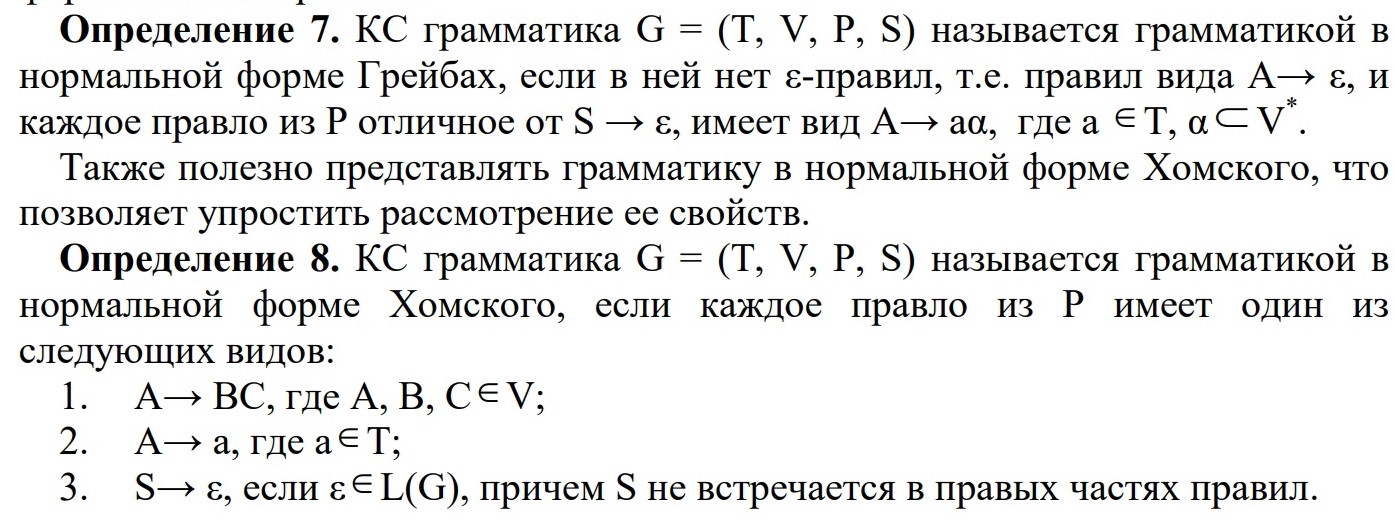
S => AB => CBB => abb

**6. Определить форму КС-грамматики и сделать ее приведение**

6.1. Задана следующая грамматика: 28. P={S → AB, S → C, D → SC, D → d, A→ G, A→CB, G → F, A→ Ab, A→ BB, B → b, C→ a, D → ε, C → ε.}

6.2. Эта же грамматика в приведённом виде: G’ = (T, V, P’,S), где P’ = {S → AB, A→CB, A→ bA’, A’→ b, A’→ bA’, A→ BB, B → b, C→ a, C→ A}

**Заменим A’ на K:** P’ = {S → AB, A→CB, A→ bK, K→ b, K→ bK, A→ BB, B → b, C→ a}



Полученная грамматика не соответствует данным определениям.

Например в ней присутствуют цепочки вида A→ bK и A→ BB.

S => AB => CBB => abb

S => AB => bKB => bbKB ... правая рекурсия  
**7. Спроектировать МП автомат для приведенной КС-грамматики**

Используя алгоритм 3.8. получим: МП = ({q}, {a,b}, {a,b,S,A,B,C,K}, δ, q0, S, {q}), в котором функция переходов δ определяется следующим образом:

1. δ(q0, ε, S) = {(q, AB)};

2. δ(q, ε, A) = {(q, BB), (q, CB), (q, bK)};

3. δ(q, ε, K) = {(q, b), (q, bK)};

4. δ(q, ε, B) = {(q, b)};

5. δ(q, ε, C) = {(q, a)};

6. δ(q, a, a) = {(q, ε)};

7. δ(q, b, b) = {(q, ε)};

* (q0, abb, S) |-1 (q, abb, AB) |-2.2 (q, abb, CBB) |-5 (q, abb, aBB) |-6 (q, bb, BB) |-4 (q, bb, bB) |-7 (q, b, B) |-4 (q, b, b) |-7 (q, ε, ε)
* (q0, bbb, S) |-1 (q, bbb, AB) |-2.1 (q, bbb, BBB) |-4 (q, bbb, bBB) |-7 (q, bb, BB) |-4 (q, bb, bB) |-7 (q, b, B) |-4 (q, b, b) |-7 (q, ε, ε)

**8. Реализовать МП автомат для приведенной КС-грамматики**

case "4.1":

Console.WriteLine("Выберите КС-грамматику: ");

/\*

\* P’ = {S → AB, A→CB, A→ bK, K→ b, K→ bK, A→ BB, B → b, C→ a}

\*/

myGrammar kcGr1 = new myGrammar(new ArrayList() { "a", "b" },

new ArrayList() { "S", "A", "B", "C", "K" },

"S");

kcGr1.AddRule("S", new ArrayList() { "A", "B" });

kcGr1.AddRule("A", new ArrayList() { "C", "B" });

kcGr1.AddRule("A", new ArrayList() { "B", "B" });

kcGr1.AddRule("C", new ArrayList() { "a" });

kcGr1.AddRule("B", new ArrayList() { "b" });

kcGr1.AddRule("A", new ArrayList() { "b", "K" });

kcGr1.AddRule("K", new ArrayList() { "b", "b","K" });

Console.Write("Debug KC-Grammar ");

kcGr1.DebugPrules();

string ans = "y";

while (ans == "y")

{

Console.WriteLine("Введите 1, 2 или 3");

switch (Console.ReadLine())

{

case "1":

myMp mpA1 = new myMp(new ArrayList() { "q0", "q1" },

new ArrayList() { "a", "b" },

new ArrayList() { "a", "b", "S", "A", "B", "C", "K"},

"q0",

"S",

new ArrayList() { "q1" });

mpA1.addDeltaRule("q0", "", "S", new ArrayList() { "q1" }, new ArrayList() { "A","B"});

mpA1.addDeltaRule("q1", "", "A", new ArrayList() { "q1" }, new ArrayList() { "C","B" });

mpA1.addDeltaRule("q1", "", "A", new ArrayList() { "q1" }, new ArrayList() { "B","B" });

mpA1.addDeltaRule("q1", "", "A", new ArrayList() { "q1" }, new ArrayList() { "b","K" });

mpA1.addDeltaRule("q1", "", "B", new ArrayList() { "q1" }, new ArrayList() { "b" });

mpA1.addDeltaRule("q1", "", "C", new ArrayList() { "q1" }, new ArrayList() { "a"});

mpA1.addDeltaRule("q1", "", "K", new ArrayList() { "q1" }, new ArrayList() { "b" });

mpA1.addDeltaRule("q1", "", "K", new ArrayList() { "q1" }, new ArrayList() { "b","K" });

mpA1.addDeltaRule("q1", "a", "a", new ArrayList() { "q1" }, new ArrayList() { "" });

mpA1.addDeltaRule("q1", "b", "b", new ArrayList() { "q1" }, new ArrayList() { "" });

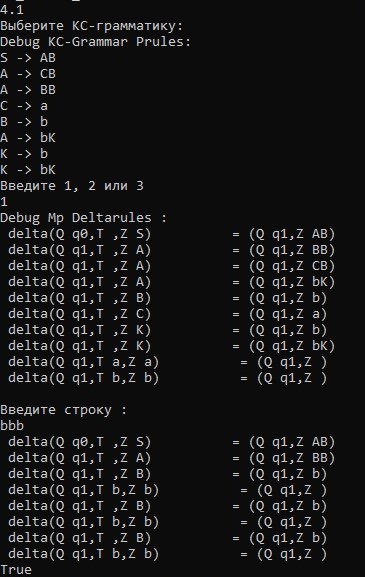
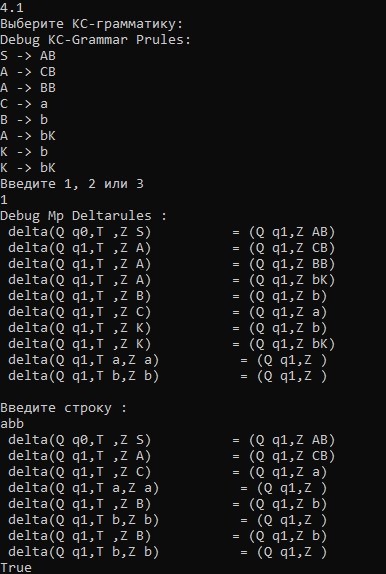
Console.Write("Debug Mp ");

mpA1.debugDelta();

Console.WriteLine("\nВведите строку :");

Console.WriteLine(mpA1.Execute(Console.ReadLine()).ToString());

break;



24. G = {i, j, \* , :,(,)}, V={S, F, L}, P = {S→ F: L, S→ (L), F→ L\*, F→ i, L→j}

**9. Для LL(1) анализатора построить управляющую таблицу M**

КС-грамматика G = {i, j, \* , :,(,)}, V={S, F, L}, P = {S→ F: L, S→ (L), F→ L\*, F→ i, L→j}

P:

1. S→ F:L,
2. S→ (L),
3. F→ L\*,
4. F→ i,
5. L→j
6. Для p1: S → F:L ; FIRST(F:L) = FIRST(F) = {i, j};
7. Для p2: S → (L) ; FIRST((L)) = {(};
8. Для p3: F → L\* ; FIRST(L\*) = {j};
9. Для p4: F → i ; FIRST(i) = {i};
10. Для p5: L → j ; FIRST(j) = {j};

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | i | j | \* | : | ( | ) | ε |
| S | F:L, 1 | F:L, 1 |  |  | (L), 2 |  |  |
| F | i, 4 | L\*, 3 |  |  |  |  |  |
| L |  | j, 5 |  |  |  |  |  |
| i | выброс |  |  |  |  |  |  |
| j |  | выброс |  |  |  |  |  |
| \* |  |  | выброс |  |  |  |  |
| : |  |  |  | выброс |  |  |  |
| ( |  |  |  |  | выброс |  |  |
| ) |  |  |  |  |  | выброс |  |
| \_|\_ |  |  |  |  |  |  | допуск |

**10. Аналитически написать такты работы LL(1) анализатора для выведенной цепочки.**

КС-грамматика G = {i, j, \* , :,(,)}, V={S, F, L}, P = {S→ F: L, S→ (L), F→ L\*, F→ i, L→j}

P:

1. S→ F:L,
2. S→ (L),
3. F→ L\*,
4. F→ i,
5. L→j

**Вывод цепочек:**

* S ⇨1 F:L ⇨4 i:L ⇨5 i:j
* S ⇨2 (L) ⇨5 (j)
* S ⇨1 F:L ⇨3 L\*:L ⇨5 j\*:L ⇨5 j\*:j

**Распознавание цепочек:**

* (i:j, S\_|\_, ε) |- (i:j, F:L\_|\_, 1) |- (i:j, i:L\_|\_, 14) |- (:j, :L\_|\_, 14) |- (j, L\_|\_, 14) |-   
  (j, j\_|\_, 145) |- (ε, \_|\_, 145)
* ( (j), S\_|\_, ε ) |- ( (j), (L)\_|\_, 2 ) |- ( j), L)\_|\_, 2 ) |- ( j), j)\_|\_, 25 ) |- ( ), )\_|\_, 25 ) |- ( ε, \_|\_, 25 )
* (j\*:j, S\_|\_, ε) |- (j\*:j, F:L\_|\_, 1) |- (j\*:j, L\*:L\_|\_, 13) |- (j\*:j, j\*:L\_|\_, 135) |-   
  (\*:j, \*:L\_|\_, 135) |-(:j, :L\_|\_, 135) |- (j, L\_|\_, 135) |- (j, j\_|\_, 1355) |-  
  (ε, \_|\_, 1355)

**11. Реализовать управляющую таблицу M для LL(1) анализатора.**

case "5.1": // LL Разбор

myGrammar exemple1 = new myGrammar(new ArrayList() { "i", "j", "(", ")", ":", "\*", "" },

new ArrayList() { "S", "F", "L" },

"S");

exemple1.AddRule("S", new ArrayList() { "F", ":", "L" });

exemple1.AddRule("S", new ArrayList() { "(", "L", ")" });

exemple1.AddRule("F", new ArrayList() { "i" });

exemple1.AddRule("F", new ArrayList() { "L", "\*" });

exemple1.AddRule("L", new ArrayList() { "j" });

LLParser parser1 = new LLParser(exemple1);

Console.WriteLine("Введите строку: ");

if (parser1.Parse1(Console.ReadLine()))

{

Console.WriteLine("Успех. Строка соответствует грамматике.");

Console.WriteLine(parser1.OutputConfigure);

}

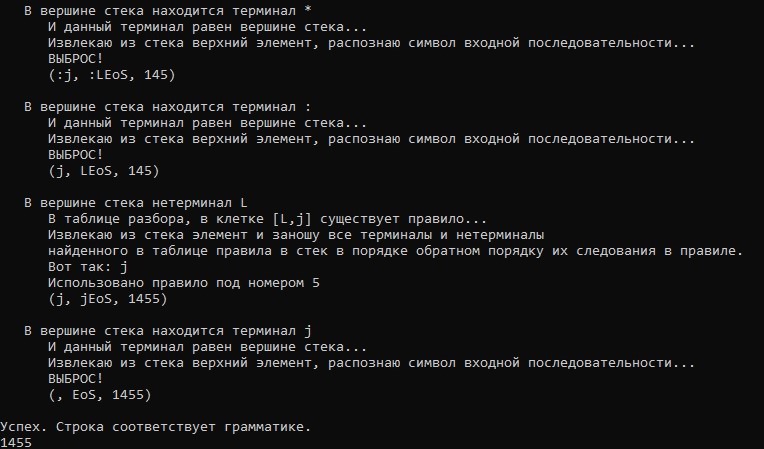
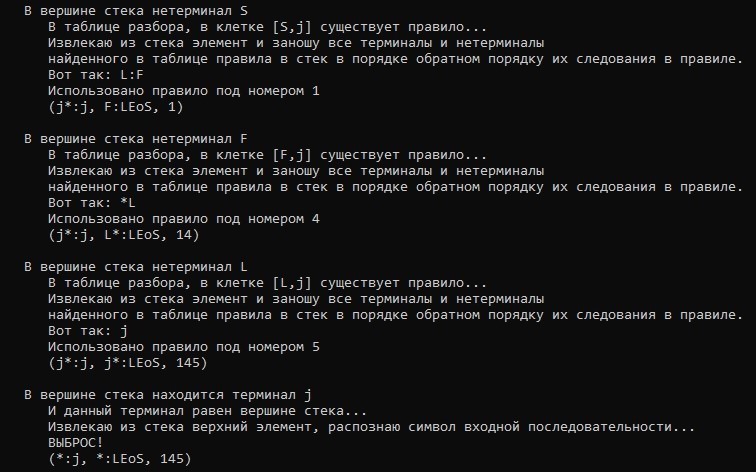
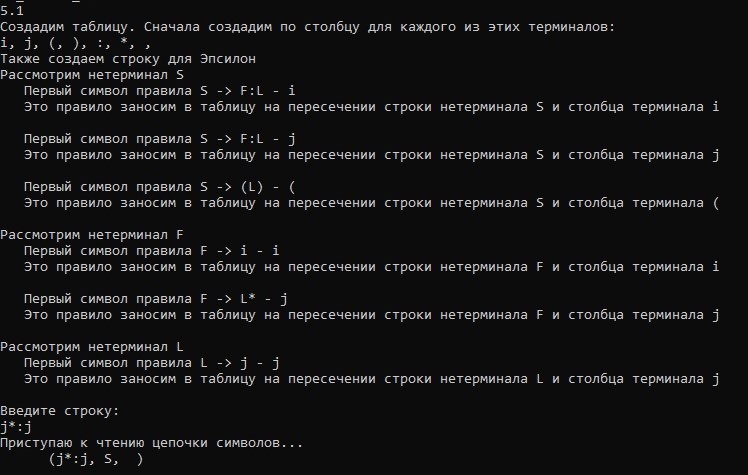
else

{

Console.WriteLine("Не успех. Строка не соответствует грамматике.");

}

break;



* **12. Построение LR(k) анализатора на основе активных префиксов и отношения OBLOW. Построить управляющую таблицу для функции перехода g(х) и действий f(u) и такты работы алгоритма.**
* **13.Реализовать LR(k)-анализатор по управляющей таблице (g,f) для LR(k) грамматики.**

24. КС-грамматика G = {i, j, \* , :,(,)}, V={S, F, L}, P = {S→ F: L, S→ (L), F→ L\*, F→ i, L→j}

P:

1. S→ F:L,

2. S→ (L),

3. F→ L\*,

4. F→ i,

5. L→j

Пополненная КС-грамматика:

КС-грамматика G’ = {i, j, \* , :,(,)}, V={S’, S, F, L}, P = {S’→S, S→ F: L, S→ (L), F→ L\*, F→ i, L→j}

P:

1. S’→S
2. S→ F:L,
3. S→ (L),
4. F→ L\*,
5. F→ i,
6. L→j

**Определение активных префиксов**

|  |  |  |
| --- | --- | --- |
| Символ грамматики | Магазинный символ | Кодируемая цепочка |
| S | S0 | ^S |
| F | F1 | F |
| L | L2 | F:L |
| L3 | (L |
| i | I5 | i |
| j | J6 | j |
| \* | \*4 | L\* |
| : | :1 | F: |
| ( | (3 | ( |
| ) | )3 | (L) |
| L | L4 | L |

**Отношение OBLOW для грамматических вхождений грамматики G**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | S0 | F1 | :1 | L2 | L3 | (3 | )3 | \*4 | I5 | J6 | L4 |
| S0 |  |  |  |  |  |  |  |  |  |  |  |
| F1 |  |  | 1 |  |  |  |  |  | 1 |  |  |
| :1 |  |  |  | 1 |  |  |  |  |  | 1 |  |
| L2 | 1 |  |  |  |  |  |  |  |  |  |  |
| L3 |  |  |  |  |  |  | 1 |  |  |  |  |
| (3 |  |  |  |  | 1 |  |  |  |  | 1 |  |
| )3 | 1 |  |  |  |  |  |  |  |  |  |  |
| \*4 |  |  | 1 |  |  |  |  |  |  |  |  |
| I5 |  |  | 1 |  |  |  |  |  |  |  |  |
| J6 |  |  |  |  |  |  | 1 | 1 |  |  |  |
| L4 |  |  |  |  |  |  |  | 1 |  |  |  |
| ^ | 1 | 1 |  |  |  | 1 |  |  |  |  |  |

**Управляющая таблица**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Функция действий f(u) | | | | | | | Функция перехода g(X) | | | | | | | | |
|  | i | j | \* | : | ( | ) | ^ | S | F | : | L | ( | ) | \* | i | j |
| S0 |  |  |  |  |  |  | Д |  |  |  |  |  |  |  |  |  |
| F1 | П | П | П | П | П | П |  |  |  | :1 |  |  |  |  |  |  |
| :1 | П | П | П | П | П | П |  |  |  |  | L2 |  |  |  |  | J6 |
| L2 | C(2) | C(2) | C(2) | C(2) | C(2) | C(2) | C(2) |  |  |  |  |  |  |  |  |  |
| L3 | П | П | П | П | П | П |  |  |  | :1 |  |  | )3 |  |  |  |
| (3 | П | П | П | П | П | П | П |  |  |  | L3 |  |  |  |  | J6 |
| )3 | С(3) | С(3) | С(3) | С(3) | С(3) | С(3) | С(3) |  |  |  |  |  |  |  |  |  |
| \*4 | C(4) | C(4) | C(4) | C(4) | C(4) | C(4) | C(4) |  |  | :1 | L4 |  |  |  |  |  |
| I5 | C(5) | C(5) | C(5) | C(5) | C(5) | C(5) | C(5) |  | F1 |  |  |  |  |  |  |  |
| J6 | C(6) | C(6) | C(6) | C(6) | C(6) | C(6) | C(6) |  |  |  |  |  | )3 | \*4 |  |  |
| L4 | П | П | П | П | П | П |  |  |  |  | L3 |  |  | \*4 |  | J6 |
| ^ | П | П | П | П | П | П |  | S0 | F1 |  | L4 | (3 |  |  |  | J6 |

**Распознавание цепочек:**

(┴, j\*:j┴, ε) ⊢П (┴J6, \*:j┴, ε) ⊢С (┴L4, \*:j┴, 6) ⊢П

(┴L4\*4, :j┴, 6) ⊢С (┴F1, :j┴, 46) ⊢П (┴F1:1, j┴, 46) ⊢П (┴F1:1J6,┴, 46) ⊢С (┴F1:1L2, ┴, 646) ⊢С (┴S0, ┴, 2646)

(┴, (j)┴, ε) ⊢П (┴(3, j)┴, ε) ⊢П (┴(3J6, )┴, ε) ⊢С (┴L3, )┴, 6) ⊢П (┴L3)3, ┴, 6)⊢С (┴S0, ┴, 36)

**14. Построение LR(k) анализатора на основе LR(0)-ситуаций и функций CLOSURE и GOTO. Построить управляющую таблицу для функции перехода g(х) и действий f(u)**

**15.Построить конечный автомат для переходов между ситуациями и такты работы алгоритма.**

Пополненная КС-грамматика:

КС-грамматика G’ = {i, j, \* , :,(,)}, V={S’, S, F, L}, P = {S’→S, S→ F: L, S→ (L), F→ L\*, F→ i, L→j}

P:

1. S’→S
2. S→ F:L,
3. S→ (L),
4. F→ L\*,
5. F→ i,
6. L→j

**Замыкание множества ситуаций:**

I = CLOSURE({S’ → ·S})

Шаг 0: I = { S’ → ·S }

Шаг 1: I = I ∪ { S → ·F : L, S→ ·(L) }

Шаг 2: I = I ∪ { F→ ·i, F→ ·L\* }

Шаг 3: I = I ∪ { L→ ·j }

I0 = { S’ → ·S, S → ·F : L, S→ ·(L) , F→ ·i, F→ ·L\*, L→ ·j}

**Функция переходов GOTO:**

I0 = CLOSURE({S’ → ·S}) = { S’ → ·S, S → ·F : L, S→ ·(L) , F→ ·i, F→ ·L\*, L→ ·j}

GOTO(I0, S) = { S’ → S· } = I1

GOTO(I0, F) = { S → F·: L } = I2

GOTO(I0, () = { S→ (·L), L→ ·j} = I3

GOTO(I0, L) = { F→ L·\* } = I4

GOTO(I0, i) = { F→ i· } = I5

GOTO(I0, j) = { L→ j· } = I6

GOTO(I2, :) = { S → (F : ·L, L→ ·j) } = I7

GOTO(I3, L) = { S → (L·) } = I8

GOTO(I3, j) = { L → j· } = присутствует как I6

GOTO(I4, \*) = { F → L\*· } = I9

GOTO(I7, L) = { S → S → F : L· } = I10

GOTO(I7, j) = { L → j· } =присутствует как I6

GOTO(I9, L) = { S → (L)· } = I11

**Каноническая форма множества ситуаций:**

C = {

I0 = { S’ → ·S, S → ·F : L, S→ ·(L) , F→ ·i, F→ ·L\*, L→ ·j},

I1 = { S’ → S· },

I2 = { S → F·: L },

I3 = { S→ (·L), L→ ·j},

I4 = { F→ L·\* },

I5 = { F→ i· },

I6 = { L→ j· },

I7 = { S → (F : ·L, L→ ·j) },

I8 = { S → (L·) },

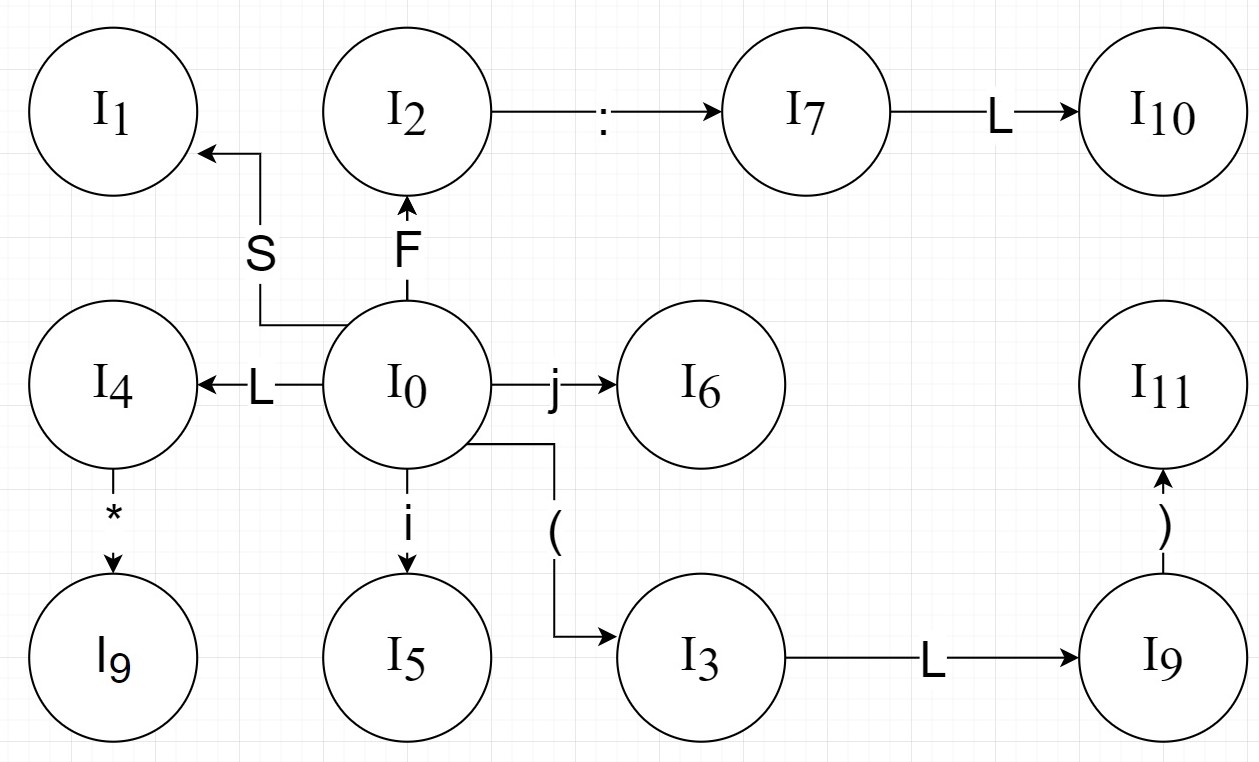
I9 = { F → L\*· }**,**

I10 = { S → S → F : L· },

I11 = { S → (L)· }

}

Диаграмма переходов автомата:



**Управляющая таблица**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **I** | **Функция действий** | | | | | | | **Функция переходов** | | |
| **i** | **j** | **\*** | **:** | **(** | **)** | ^ | **S** | **F** | **L** |
| **0** | **П(5)** | **П(6)** |  |  | **П(3)** |  |  | **1** | **2** | **4** |
| **1** |  |  |  |  |  |  | **Д** |  |  |  |
| **2** |  |  |  | **П(7)** |  |  |  |  |  | **8** |
| **3** |  | **П(6)** |  | **П, 7** |  |  |  |  |  |  |
| **4** |  |  | **П(9)** |  |  |  |  |  |  |  |
| **5** | **С(4)** | **С(4)** | **С(4)** | **С(4)** | **С(4)** | **С(4)** |  |  |  |  |
| **6** | **С(5)** | **С(5)** | **С(5)** | **С(5)** | **С(5)** | **С(5)** |  |  |  |  |
| **7** |  | **П, 6** |  |  |  |  |  |  |  | **10** |
| **8** |  |  |  | **С, 2** |  | **П(11)** |  |  |  |  |
| **9** | **С(3)** | **С(3)** | **С(3)** | **С(3)** | **С(3)** | **С(3)** |  |  |  |  |
| **10** | **С(1)** | **С(1)** | **С(1)** | **С(1)** | **С(1)** | **С(1)** |  |  |  |  |
| **11** | **С(2)** | **С(2)** | **С(2)** | **С(2)** | **С(2)** | **С(2)** |  |  |  |  |

**Распознавание цепочек:**

(0, j\*:j┴, ε) ⊢П (0 6, \*:j┴, ε) ⊢С (0 4, \*:j┴, 5) ⊢П

(0 4 9, :j┴, 5) ⊢С (0 2, :j┴, 53) ⊢П (0 2 7, j┴, 53) ⊢П (0 2 7 6, ┴, 53) ⊢С (0 2 7 10, ┴, 535) ⊢С (0 1, ┴, 5351) - ДОПУСК

S =>1 F:L =>4 L\*:L =>6 j\*:L =>6 j\*:j

P:

1. S’→S
2. S→ F:L,
3. S→ (L),
4. F→ L\*,
5. F→ i,
6. L→j

**16. . Реализовать LR(k)-анализатор по управляющей таблице (g,f) для LR(k) грамматики.**

6.1

Исходная

КС - грамматика :

Алфавит нетерминальных символов: SFL

Алфавит терминальных символов: : :\*i()j

Правила :

{ S F:L; F L\*; L j; F i; S (L) }

После удаления е-продукций

Правила:

S F:L

F L\*

L j

F i

S (L)

П S

Терминалы : :\*i()j$

Нетерминалы: SFLП

-----

Вычислены множества FIRST для символов грамматики и строк

First( S ): (ij

First( F ): ij

First( L ): j

First( П ): (ij

First( F:L ): ij

First( L\* ): j

First( j ): j

First( i ): i

First( (L) ): (

First( S ): (ij

Cоздана последовательность С:

I0 { П .S,$; S .F:L,$; S .(L),$; F .L\*,:; F .i,:; L .j,\* }

I1 { F i.,: }

I2 { S (.L),$; L .j,) }

I3 { L j.,\* }

I4 { П S.,$ }

I5 { S F.:L,$ }

I6 { F L.\*,: }

I7 { S F:.L,$; L .j,$ }

I8 { F L\*.,: }

I9 { L j.,) }

I10 { S (L.),$ }

I11 { S (L).,$ }

I12 { L j.,$ }

I13 { S F:L.,$ }

Создана ACTION таблица

ACTION[0, (] = s 2

ACTION[3, \*] = r L j

ACTION[2, j] = s 9

ACTION[1, :] = r F i

ACTION[0, j] = s 3

ACTION[7, j] = s 12

ACTION[6, \*] = s 8

ACTION[5, :] = s 7

ACTION[4, $] = a

ACTION[11, $] = r S (L)

ACTION[10, )] = s 11

ACTION[9, )] = r L j

ACTION[8, :] = r F L\*

ACTION[13, $] = r S F:L

ACTION[12, $] = r L j

ACTION[0, i] = s 1

Создана GOTO таблица

GOTO[2, L] = 10

GOTO[0, S] = 4

GOTO[7, L] = 13

GOTO[0, F] = 5

GOTO[0, L] = 6

Введите строку:

j\*:j

Введена строка: j\*:j$

Процесс вывода:

L->j

F->L\*

L->j

S->F:L

Строка допущена