Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Лабораторная работа №9-11
4 семестр

«Построение управляющей таблицы M для LL(k) анализатора. Аналитическое написание правил вывода для цепочки LL(k) анализатора. Реализация управляющей таблицы M для LL(k) анализатора»

Студент:	Шевчук П.В.
Группа:	М80-204Б
Преподаватель:	Семенов А.С.
Вариант	15
Оценка:	
Дата:	

Москва
2018

Лабораторная работа №9. Построение управляющей таблицы M для LL(k) анализатора.

Задание: 3. T = {i, : , -,(,)}, V = {S, F, L}, P = {S→ F : L, S→ (), F→ - L, F→ i, L→F}

Управляющая таблица должна содержать 10 строк, помеченных символами из множества (VT{⊥}), и 7 столбцов, помеченных символами из множества (T{ε}).

	i	-	:	(	)	ε
S	F : L,1	F : L,1		(S), 2		
F	i,3	-L, 4				
L	F,5					
i	выброс					
-		выброс				
:			выброс			
(				выброс		
)					выброс	
⊥						допуск

Шаг 1. Строим таблицу построчно. Последовательно рассмотрим все нетерминальные символы.

1. Нетерминалу S соответствует правило вывода грамматики p1: S → (F : L). Так как FIRST((F : L)) = { (}, один терминальный символ, то:

M(S, ( ) = M((F : L), 1)

Правило грамматики	Множество	Значение M
p_1) S→F : L	FIRST(F : L) = {(}	M(S,()=((F : L),1)
p_2) F→ -L	FIRST(-L) = {-}	M(F,-)=(-L,2)
p_3) F→ i	FIRST(i)={i}	M(F,i)=(i,3)

p_4) L→F	FIRST(F)={i}	M(L,i)=(F,4)

Шаг 2. Далее всем элементам таблицы, находящимся на пересечении строки и столбца, отмеченных одним и тем же терминальным символом, присвоим значение ВЫБРОС.

Шаг 3. Элементу таблицы M(⊥, ε) присвоим значение ДОПУСК.

Шаг 4. Остальным элементам таблицы присвоим значение ОШИБКА и представим результат в виде таблицы.

Начальное содержимое магазина - $S_⊥$

Лабораторная работа №10. Аналитическое написание правил вывода для цепочки LL(k) анализатора.

Рассмотрим работу вывода для цепочки (-i:i)

Текущая конфигурация	Значение M
((-i:i), S⊥,Ɛ) ⊢	M(S,i)=((S), 2)
((-i:i), (S)⊥, 2) ⊢	M(S,()=(F:L,1)
((-i:i), (F:L)⊥, 21) ⊢	M((,( ) = ВЫБРОС
(-i:i), F:L)⊥, 21) ⊢	M(F,-)=(-L,3)
(-i:i), -L:L)⊥, 23) ⊢	M(-,-)=ВЫБРОС
(i:i), L:L)⊥, 213) ⊢	M(L,i)=(F,5)
(i:i), F:L)⊥, 2135) ⊢	M(F,i)=(F,4)
(i:i), i:L)⊥, 2135) ⊢	M(i,i)=ВЫБРОС

(:i), :L)⊥, 2135) ⊢ (i), L)⊥, 2135) ⊢ (i), i)⊥, 21354) ⊢ ( ),)⊥, 2135454) ⊢ (Ɛ, ⊥,21354 54) ⊢	M(:,:)=ВЫБРОС M(L,i)=(F,4) M(i,i)=ВЫБРОС M(),))=ВЫБРОС M(⊥,Ɛ)=ДОПУСК

C) Определить является ли LL(k)-грамматика сильно LL(k)-грамматикой

Если в КС-грамматике G = (T, V, P, S) для двух различных A-правил A→ β и A→ γ выполняется:

FIRSTk(β FOLLOWk (A)) ∩ FIRSTk(γ FOLLOWk (A)) = Ø, то такая

КС-грамматика называется сильно LL(k)-грамматикой.

Пусть задана LL(2)-грамматика G с правилами {S→ aAaa | bAba, A → b | ε} множество FOLLOW2(A) = { aa, ba}, а значит, для A-правила:

FIRST2(b FOLLOW2 (A)) ∩ FIRST2(γ FOLLOW2 (A)) = {ba}, и это не сильно LL(2)-грамматика

Лабораторная работа №11. Реализация управляющей таблицы M для LL(k) анализатора

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;

namespace LL_1__анализатор

{

    class Program

    {

        static void Main(string[] args)

        {

            LL_analyzer analyzer = new LL_analyzer();

            string line;
```

```
string answer;
analyzer.Set_M();
while (true)
{
    Console.WriteLine("Введите строчку для анализа:");
    line = Console.ReadLine();
    analyzer.set_entrance_line(line);
    if (analyzer.analysis())
    {
        Console.WriteLine("Строка принадлежит языку!!!");
        Console.WriteLine("Переходы:" + analyzer.output_line);
        Console.WriteLine();
    }
    else
    {
        Console.WriteLine("Строка не принадлежит языку!!!");
        Console.WriteLine("Переходы:" + analyzer.output_line);
        Console.WriteLine();
    }
    Console.WriteLine("Проверить ещё одну строчку?");
    analyzer.output_line = null;
    answer = Console.ReadLine();
    //analyzer.set_entrance_line(line);
    if (answer != "y" && answer != "yes")
    {
        if (answer == "no" || answer == "n")
        {
            break;
        }
        else
        {
            if (answer != "yes" && answer != "y")
            {
```

```
                            Console.WriteLine("Некорректный ввод!!!");
                            Console.ReadLine();
                            break;
                        }

                    }
                }//if

                analyzer.MP.Push("_");
                analyzer.MP.Push("S");
            }//while
        }//Main
    }
}
class LL_analyzer //LL-анализатор
{
    ArrayList Q = new ArrayList(); //мн-во всех возможных состояний
    ArrayList V = new ArrayList(); //мн-во всех нетерминальных символов
    ArrayList T = new ArrayList(); //мн-во всех терминальных символов
    ArrayList P = new ArrayList(); //мн-во всех правил вывода
    string entrance_line; //входная строка (на ленте)

    public void set_entrance_line(string line)
    { entrance_line = line; }

    public string output_line; //выходная строка (на ленте)
    char Current_state; //текущее состояние автомата
    public Stack MP = new Stack(); //стэк моделирует Магазинную память
    int[,] M; //управляющая таблица

    public LL_analyzer() //конструктор
    {
```

```
        MP.Push("_"); //"дно" стека
        MP.Push("S"); //S - аксиома грамматики

        V.Add("S");
        V.Add("F");
        V.Add("L");

        T.Add("i");
        T.Add("-");
        T.Add(":");
        T.Add("(");
        T.Add(")");

        Rule rule;
        rule = new Rule("S", "F:L");
        P.Add(rule);
        rule = new Rule("S", "(S)");
        P.Add(rule);
        rule = new Rule("F", "-L");
        P.Add(rule);
        rule = new Rule("F", "i");
        P.Add(rule);
        rule = new Rule("L", "F");
        P.Add(rule);

        output_line = null;

    } //конец коструктора

    public void Set_M() //заполнение управляющей таблицы
    {
        M = new int[3, 5];
        for (int i = 0; i < 3; i++)
```

```
            for (int j = 0; j < 5; j++)
                M[i, j] = 0;
        M[0, 0] = 1; M[1, 0] = 4; M[2, 0] = 5; M[0, 1] = 1; M[1, 1] = 3; M[2, 1] = 5;
M[0, 3] = 2;
    } //конец заполнения упр. табл.

    public bool analysis()
    {
        int i = 0;
        char c;
        int k = 0;
        int m = 0;
        char[] str;
        string to_stack;
        Current_state = 'S';
        int length = entrance_line.Length;
        while (i < length)
        {
            if ((String)MP.Peek() == "_" && i == length)
            {
                return true;
            }
            else
            {
                str = MP.Peek().ToString().ToCharArray();
                c = str[0];
                if (c >= 'A' && c <= 'Z') //Если на вершине нетерминал
                {
                    if (T.IndexOf(entrance_line.Substring(i, 1)) == -1) //строка состоит
из символов не принадлежащих языку
                    {
                        return false;
                    }
```

```
                    else

                    {

                        to_stack = c.ToString();

                        k = V.IndexOf(to_stack); /*номер строки, которая нам нужна в
таблице

(нетерминал в стеке)*/

                        m = T.IndexOf(entrance_line.Substring(i, 1)); /*номер столбца,
который нам нужен в таблице

(терминал на ленте)*/

                        //Console.WriteLine(M[k, m]);

                        to_stack = ((Rule)P[M[k, m] - 1]).Get_RightTerm();


                        int l = to_stack.Length;

                        MP.Pop(); //выбрасываем из стека левую часть применяемого правила

                        for (int d = l - 1; d >= 0; d--) //кладём в стек в нужном порядке

                        { //правую часть применяемого правила

                            MP.Push((to_stack.Substring(d, 1)).ToString());

                        }

                        output_line = output_line + M[k, m]; //формирование выходной
строки


                    } //(конкатенация строк)

                }

                else

                {

                    if (MP.Peek().ToString() == entrance_line.Substring(i, 1)) //на
верхушке стека и текущий символ на ленте одинаковы

                    {

                        MP.Pop(); i++; //выбрасываем с верхушки стека и просматриваем
следующий символ на ленте

                        //str = entrance_line.ToCharArray();

                        //Current_state = str[i];

                    }


                    else
```

```
                {
                    return false;
                }

            }//else
        }//else
    }//while
    if (i == length && (String)MP.Peek() != "_")
    {
        return false;
    }
    else
    {
        return true;
    }

}//конец LL_analyzer

class Rule
{
    private string LeftTerm = null;
    private string RightTerm = null;

    public string Get_LeftTerm() { return LeftTerm; }
    public string Get_RightTerm() { return RightTerm; }

    public Rule(string LeftTerm, string RightTerm)
    {
        this.LeftTerm = LeftTerm;
        this.RightTerm = RightTerm;
    }
}//конец Rule
}
```

Вывод программы:

Введите строчку для анализа:

(-i:i)

Строка принадлежит языку!!!

Переходы:2135454

Проверить ещё одну строчку?