Минобрнауки России

ФГБОУ ВПО «НИУ МЭИ» АВТИ

Кафедра математического и компьютерного моделирования

**Лабораторная работа №3**

**Математическое обеспечение ЭВМ**

**«Построение синтаксического анализатора»**

**Работу выполнил:**

Солонин Егор А-14-19

Вариант 14

**Работу принял:**

Князев А. В.

Москва 2021

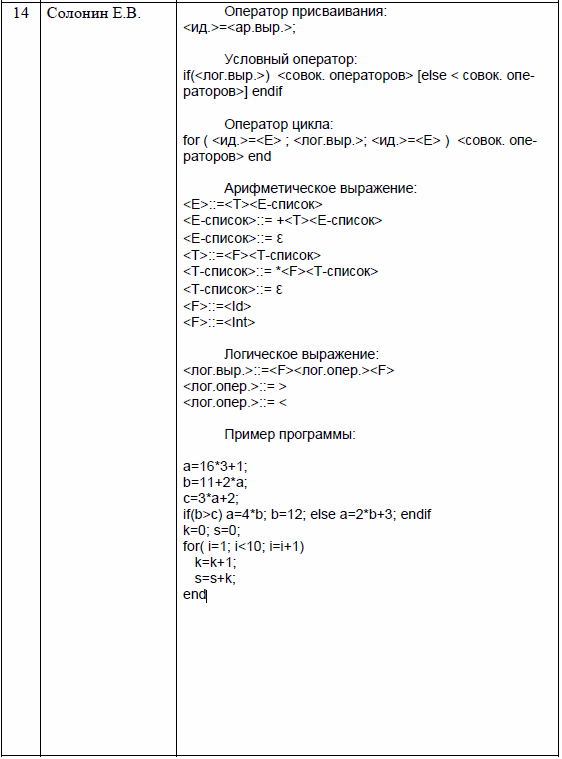
**Задание:**

1. Преобразовать заданную грамматику в LL(1)-грамматику.

2. Разработать МП-автомат для нисходящего грамматического разбора предложений данного языка.

3. Разработать функцию, реализующую МП-автомат.

4. Разработать программу, иллюстрирующую работу созданного автомата для данного языка. Программа разрабатывается как приложение с графическим интерфейсом на языке C# в среде Visual Studio. Не должны использоваться коллекции. Не должны использоваться регулярные выражения и другие средства разбора строк.

*Индивидуальный вариант (№14):*

**LL(1)-грамматика:**

1. <программа>::= <оператор><спис. операторов>
2. <спис. операторов>::= <оператор><спис. операторов>
3. <программа>::=
4. <оператор>::= <идент.>=<E>;
5. <оператор>::=for(<идент.>=<E>; <лог.выр.>; <идент.>=<E>) <спис. операторов> end
6. <оператор>::= if(<лог.выр.>) <спис. операторов> <иначе> endif
7. <иначе>::= else <спис. операторов>
8. <иначе>::=
9. <лог.выр.>::= <F><лог.опер.><F>
10. <лог.опер.>::= >
11. <лог.опер.>::= <
12. <E>::= <T><E-список>
13. <E-список>::= +<T><E-список>
14. <E-список>::=
15. <T>::= <F><T-список>
16. <T-список>::=  \*<F><T-список>
17. <T-список>::=
18. <F>::= <идент.>
19. <F>::= <целое>

**Множества выбора:**

Выбор(1) = {<идент.>, if, for}

Выбор(2) = {<идент.>, if, for}

Выбор(3)= {“-|”}

Выбор(4)= {<идент.>}

Выбор(5)= { for  }

Выбор(6)= {if}

Выбор(7)= {else}

Выбор(8)= {“-|”}

Выбор(9)= {<идент.>, <целое>}

Выбор(10)= {>}

Выбор(11)= {<}

Выбор(12)= {<идент.>, <целое>}

Выбор(13)= {+}

Выбор(14)= {;}

Выбор(15)= {<идент.>, <целое>}

Выбор(16)= {\*}

Выбор(17)= {+,;}

Выбор(18)= {<идент.>}

Выбор(19)= {<целое>}

**Управляющая таблица МП-автомата:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Магаз. символы | <идент> | <целое> | for | if | else | + | \* | ( | ) | > | < | = | ; | -| |
| <программа> | 1 | O | 1 | 1 | O | O | O | O | O | O | O | O | O | O |
| <спис. операторов> | 2 | O | 2 | 2 | O | O | O | O | O | O | O | O | O | 3 |
| <оператор> | 4 | O | 5 | 6 | O | O | O | O | O | O | O | O | O | O |
| <else> | 8 | O | 8 | 8 | 7 | O | O | O | O | O | O | O | O | 8 |
| <лог.выр.> | 9 | 9 | O | O | O | O | O | O | O | O | O | O | O | O |
| <лог. опер.> | O | O | O | O | O | O | O | O | O | 10 | 11 | O | O | O |
| <E> | 12 | 12 | O | O | O | O | O | O | O | O | O | O | O | O |
| <E список> | O | O | O | O | O | 13 | O | O | O | O | O | O | 14 | O |
| <T> | 15 | 15 | O | O | O | O | O | O | O | O | O | O | O | O |
| <T список> | O | O | O | O | O | 16 | 16 | O | O | O | O | O | 17 | O |
| <F> | 18 | 19 | O | O | O | O | O | O | O | O | O | O | O | O |
| <идент.> | # | O | O | O | O | O | O | O | O | O | O | O | O | O |
| = | O | O | O | O | O | O | O | O | O | O | O | # | O | O |
| ( | O | O | O | O | O | O | O | # | O | O | O | O | O | O |
| ) | O | O | O | O | O | O | O | O | # | O | O | O | O | O |
| ; | O | O | O | O | O | O | O | O | O | O | O | O | # | O |
| if | O | O | O | O | # | O | O | O | O | O | O | O | O | O |
| for | O | O | O | O | O | O | O | O | O | O | O | O | O | O |
| ˅ | О | О | О | О | О | О | О | О | О | О | О | O | O | Д |

Начальное содержимое магазина: <программа>˅

1. Заменить(<оператор><спис. операторов>), Держать
2. Заменить(<оператор><спис. операторов>), Держать
3. Вытолкнуть, Держать
4. Заменить(=<E>;), Сдвиг
5. Заменить(for( <идент.>=<E>; <лог.выр.>; <идент.>=<E>) <спис. операторов> end), Сдвиг
6. Заменить((<лог. выр.>) <спис. операторов><else>), Сдвиг
7. Заменить(<оператор>), Сдвиг
8. Вытолкнуть, Держать
9. Заменить(<F><лог.опер.><F>), Держать
10. Вытолкнуть, Сдвиг
11. Вытолкнуть, Сдвиг
12. Заменить(<T><E-список>), Держать
13. Заменить(<T><E-список>), Сдвиг
14. Вытолкнуть, Держать
15. Заменить(<F><T-список>), Держать
16. Заменить(<F><T-список>), Сдвиг
17. Вытолкнуть, Держать
18. Вытолкнуть, Сдвиг
19. Вытолкнуть, Сдвиг

# - Вытолкнуть, Сдвиг

Д – Допустить

О - Отвергнуть

**Алгоритм работы синтаксического анализатора на псевдокоде:**

**Описание интерфейса программы:**

**Алгоритмы операций на псевдокоде:**

class LexBlock – класс лексического блока, содержит:

* enum LexType – перечисление всех лексем
* char[] Terminal – все терминальные символы
* LinkedListNode table – таблица идентификаторов
* int index – свойство, указывает на текущий индекс в строке
* TerminalExists(char item) - функция, устанавливает принадлежность символа терминалу
* isLetter(char l) – функция, устанавливает является ли символ буквой
* GetLexem(string data)– функция, позволяющая получить значение лексемы
* LexType GetLexemType(string lex) – функция, позволяющая получить тип лексемы

class Form1 : Form - содержит:

* void FillTable\_Lexems() – процедура, заполняющая таблицу лексем
* void FillTable\_Ids() – процедура, заполняющая таблицу идентификаторов

GetLexem(string data):

1. Если встречаем не лексему – переходим к следующему символу
2. Если дошли до конца строки – возвращаем \0
3. Идем по строке начиная с index
   1. Если встретили терминальный символ
      1. Если i == index – возвращаем текущий символ из строки
      2. Иначе возвращаем подстроку – все что идет после index до терминального символа
   2. Если дошли до конца строки, то возвращаем подстроку все что идет после index до конца строки
4. Возвращаем \0

LexType GetLexemType(string lex):

Возвращаем соответствующий тип лексемы. Отдельно обрабатываем случаи для чисел, не валидных символов и идентификаторов

void FillTable\_Lexems():

1. Получаем очередную лексему Lexem пока не дошли до конца строки.
2. Получаем ее тип(атрибут)
3. Добавляем новую строчку в dataDridView(таблицу лексем)
4. Если (тип атрибута – целое или идентификатор) и (таблица идентификаторов LinkedListNode table не содержит Lexem), то:
   1. Создаем экземпляр идентификатора
   2. Добавляем его в LinkedListNode table.

void FillTable\_Ids():

Идем по LinkedListNode table и добавляем каждую запись в dataDridView(таблицу идентификаторов)

**Тесты работы программы:**

**Листинг программы:**