МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №1

по дисциплине: Теория автоматов и формальных языков
тема: «Формальные грамматики. Выводы»

Выполнил: ст. группы ПВ-223
Игнатьев Артур Олегович
Проверил:
Рязанов Юрий Дмитриевич

Белгород 2024 г.

Цель работы: изучить основные понятия теории формальных языков и грамматик.

Вариант 3

1. КС-грамматика
   1. S→Ssa
   2. S→b
   3. S→Ab
   4. A→AaA
   5. A→BaB
   6. A→S
   7. B→b
   8. B→aA

2. Последовательности правил вывода
   1. 1, 2, 3, 4, 5, 6, 7, 8, 6, 2, 2
   2. 1, 2, 3, 4, 5, 7, 8, 6, 2, 6, 2
   3. 1, 3, 4, 6, 2, 5, 8, 6, 2, 7, 2
   4. 1, 2, 3, 4, 6, 6, 2, 2, 7, 8, 6

Работа выполнялась на языке программирования Java. Облегчающих вычисления библиотек при выполнении работы не использовалось.

Использовалась библиотека **Scanner** для возможности вводить данные с клавиатуры. Библиотеки **ArrayList** и **Arrays** использовались для работы с массивами и коллекциями.

Задание 1. Написать программу выполняющую левый вывод в заданной КС-грамматике

Код программы:

```
//Левый вывод
public static String leftOutput(ArrayList<String> rules, Scanner in) {
      String startChain = "S";
      String intermediateChain = startChain;
      ArrayList<Integer> currentRules = new ArrayList<>(10000);

      int step = 1;

      while (!intermediateChain.equals(intermediateChain.toLowerCase()))
      {
          System.out.println("Шаг " + step);
          System.out.println("Текущая цепочка: " + intermediateChain);
          StringBuilder newIntermediateChain = new StringBuilder();
          ArrayList<Integer> applicableRules = new ArrayList<>();

          for (char c : intermediateChain.toCharArray()) {

          // Если символ верхнего регистра, пытаемся найти правила для
                                                                  замены
              if (Character.isUpperCase(c)) {
                  System.out.println("\nМожно применить: ");

                  // Показываем доступные правила
                  for (int j = 0; j < rules.size(); j++) {
                      String currentRule = rules.get(j);
                      if (currentRule.charAt(0) == c) {
                          applicableRules.add(j + 1); // Сохранение
                                                      номера правила
                          System.out.println((j + 1) + ": " +
                                                       currentRule);
                      }
                  }

                  if (applicableRules.isEmpty()) {
                      System.out.println("Правило не найдено, символ
                                          останется без изменений.");
                      newIntermediateChain.append(c);  // Оставляем
                                                    символ без изменений
                  } else {
```

```
                    System.out.println("Применим следующее правило:
                                                                    ");
                    int numberSelectRule = in.nextInt();
                    currentRules.add(numberSelectRule);
                    String replaceString = rules.get(numberSelectRule
                                                - 1).substring(2);
                    newIntermediateChain.append(replaceString); //
                                                    Заменяем символ
                }
            }else {
                newIntermediateChain.append(c); // Простой символ
                                                            добавляем
            }
        }

        // Обновляем интермедиатчную цепочку
        intermediateChain = newIntermediateChain.toString();
        step++;
    }

    System.out.println();
    System.out.println("Последовательность правил:" + currentRules);
    System.out.println();

    String tree = buildOutputTree(currentRules, rules);
    System.out.println("Дерево вывода в линейно скобочной форме:"  +
                                                    buildOutputTree)

    return intermediateChain;
}

//Построение дерева вывода в линейно скобочной форме
public static String buildOutputTree(ArrayList<Integer> currentRules,
ArrayList<String> rules) {
    StringBuilder outputTree = new StringBuilder();

    // Проходим по последовательности примененных правил
    for (int ruleIndex : currentRules) {
        String rule = rules.get(ruleIndex - 1); // Получаем
                                соответствующее правило (с учетом
                                                        индексации)
        char lhs = rule.charAt(0); // Левая часть правила (нетерминал)
        String rhs = rule.substring(2); // Правая часть правила

        outputTree.append(lhs); // Добавляем нетерминал
        outputTree.append("("); // Открывающая скобка для продукции
```

```
            // Собирать строки для терминалов
            StringBuilder terminals = new StringBuilder();

            // Проходим по символам правой части
            for (char productionChar : rhs.toCharArray()) {
                if (Character.isUpperCase(productionChar)) {
                    // Если это нетерминал, просто добавляем его в
                                                                    результат
                    outputTree.append(productionChar);
                } else {
                    // Если это терминал, добавляем его к списку
                                                                    терминалов
                    terminals.append(productionChar).append(","); //
                                        Используем запятую для разделения
                }
            }

            // Удаляем последнюю запятую в строке терминалов, если есть
                                                                        они
            if (terminals.length() > 0) {
                terminals.setLength(terminals.length() - 1); // Убираем
                                                    последнюю запятую
            }

            // Добавляем терминалы, если они есть
            if (terminals.length() > 0) {
                outputTree.append(terminals);
            }

            outputTree.append(")"); // Закрывающая скобка для продукции
        }

        return outputTree.toString(); // Возвращаем строку представления
                                                                    дерева
    }
```

Результат работы программы:

```
Введите количесвто правил:
8
Введите 1 правило:
S-Ssa
Введите 2 правило:
S-b
Введите 3 правило:
S-Ab
Введите 4 правило:
A-AaA
Введите 5 правило:
A-BaB
Введите 6 правило:
A-S
Введите 7 правило:
B-b
Введите 8 правило:
B-aA
```

Выполнение левого вывода:

```
Шаг 1
Текущая цепочка: S

Можно применить:
1: S-Ssa
2: S-b
3: S-Ab
Применим следующее правило:
3
Шаг 2
Текущая цепочка: Ab

Можно применить:
4: A-AaA
5: A-BaB
6: A-S
Применим следующее правило:
6
```

```
Шаг 3
Текущая цепочка: Sb

Можно применить:
1: S-Ssa
2: S-b
3: S-Ab
Применим следующее правило:
2

Последовательность правил:[3, 6, 2]

Полученная цепочка: bb
Дерево вывода в линейно скобочной форме: S(A(S(b))b)
```

Задание 2

Выполнить левый (правый вывод) терминальной цепочки в заданной грамматике, построить дерево вывода. Определить, существует ли неэквивалентный вывод полученной цепочки и, если существует, представить его деревом вывода.

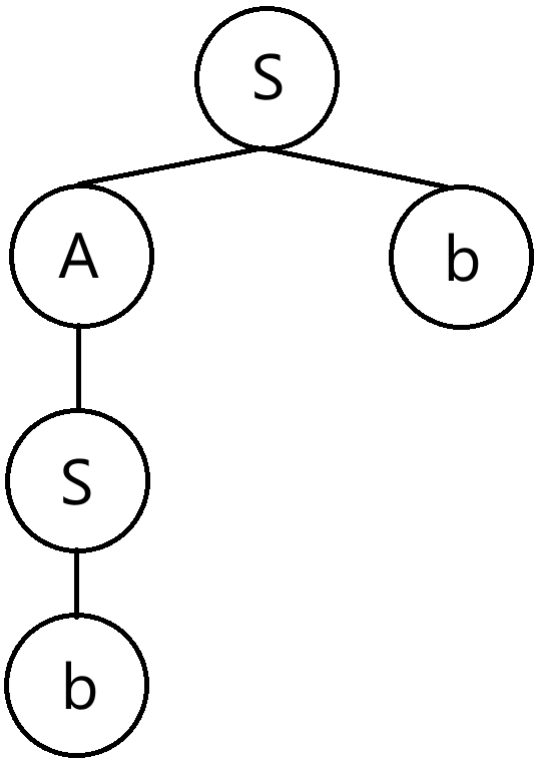
Выполним левый вывод терминальной цепочки: $\underset{3}{S} \rightarrow \underset{6}{Ab} \rightarrow \underset{2}{Sb} \rightarrow bb$

Терминальная цепочка: bb

Последовательность правил: 3 6 2

Дерево вывода (ЛСФ): S(A(S(b)b)

Дерево вывода:



Неэквивалентных выводов полученной терминальной цепочки быть не может, так как данная КС-грамматика также может называться **неукорачивающей грамматикой.**

1. Следовательно, чтобы получить терминальную цепочку длинны 2, “S” можно преобразовать только по правилу 3(S → Ab), так как правило под номером 1(S → Ssa) и правило 2(S → b) не даст получить впоследствии цепочку длинны 2.

2. Получив промежуточную цепочку “Ab”, ее можно преобразовать только по правилу 6(A → S), так как правила под номерами 4(A → AaA) и 5(A → BaB) — увеличивают длину цепочки и это так же не позволит в последствии получить терминальную цепочку длинны 2.

3. Получив промежуточную цепочку “Sb”, аналогично первому пункту можно заметить что правила 1(S → Ssa) и правило 3(S → Ab) увеличивает длину цепочки, что не позволит нам получит цепочку длинны 2.

Найдем цепочку для которой существуют неэквивалентные вывод и запишем их:

Найденная цепочка: babb

Для данной цепочки существует неэквивалентный вывод. Запишем их и построим деревья выводов.

1-й вывод:

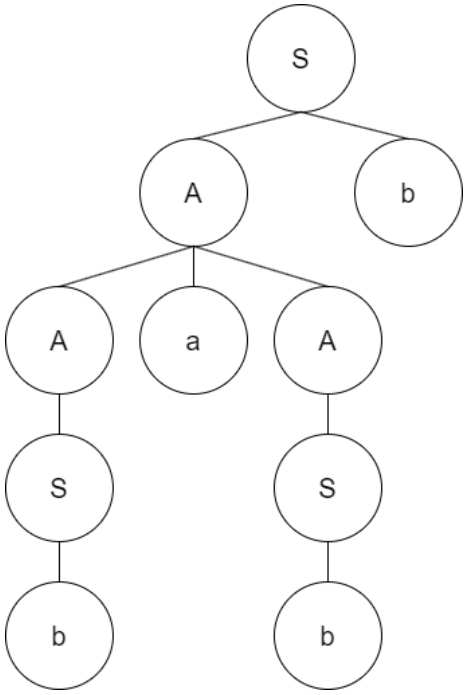
$$\underset{3}{S} \rightarrow \underset{4}{Ab} \rightarrow \underset{6}{AaAb} \rightarrow \underset{2}{SaAb} \rightarrow \underset{6}{baAb} \rightarrow \underset{2}{baSb} \rightarrow babb$$

2-й вывод:

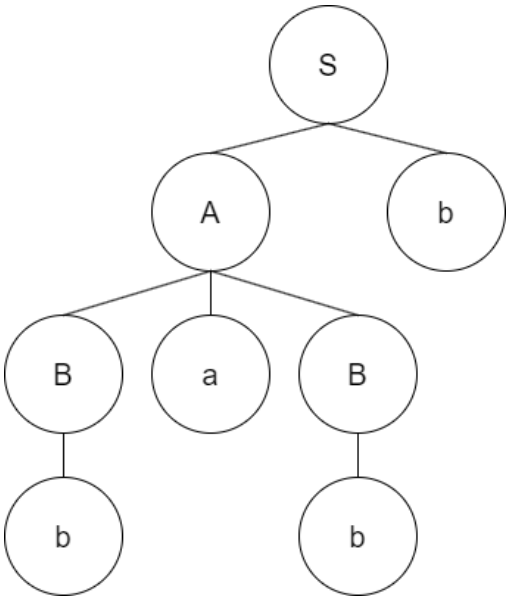
$$\underset{3}{S} \rightarrow \underset{5}{Ab} \rightarrow BaBb \rightarrow \underset{7}{Babb} \rightarrow babb$$

Деревья выводов:

Для левого вывода:



Для правого вывода:



Цепочка babb получается при применении последовательности правил 3, 4, 6, 2, 6, 2; а так же при 3, 5, 7, 7. Данным выводам соответствуют различные деревья, значит выводы не эквивалентны.

Задание 3

Написать программу, определяющую, можно ли применить заданную последовательность правил при левом выводе цепочки в заданной КС-грамматике:

Код программы:

```
// Задание 3
    public static boolean leftOutputWithSequenceCommand(ArrayList<String>
rules, Scanner in, int[] commands) {
        String startChain = "S";
        String intermediateChain = startChain;
        ArrayList<Integer> currentRules = new ArrayList<>(10000);
        int step = 1;
        int commandIndex = 0; // Индекс для отслеживания продвигающегося
                                                    по массиву команд

        while (!intermediateChain.equals(intermediateChain.toLowerCase()))
{
            System.out.println("Шаг " + step);
            System.out.println("Текущая цепочка: " + intermediateChain);
            StringBuilder newIntermediateChain = new StringBuilder();
            ArrayList<Integer> applicableRules = new ArrayList<>();

            for (char c : intermediateChain.toCharArray()) {
                // Если символ верхнего регистра, пытаемся найти правила
                                                        для замены
                if (Character.isUpperCase(c)) {
                    System.out.println("\nМожно применить: ");

                    // Показываем доступные правила
                    for (int j = 0; j < rules.size(); j++) {
                        String currentRule = rules.get(j);
                        if (currentRule.charAt(0) == c) {
                            applicableRules.add(j + 1); // Сохранение
                                                    номера правила
                            System.out.println((j + 1) + ": " +
                                                     currentRule);
                        }
                    }

                    if (applicableRules.isEmpty()) {
                        System.out.println("Правило не найдено, символ
```

```
                                        останется без изменений.");
                    newIntermediateChain.append(c); // Оставляем
                                                символ без изменений
                } else {
                    int numberSelectRule;
                    // Если имеются команды для выполнения, используем
                                                                    их
                    if (commandIndex < commands.length) {
                        numberSelectRule = commands[commandIndex++];
                    } else {
                        System.out.println("Выберите правило:");
                        numberSelectRule = in.nextInt();
                    }
                    currentRules.add(numberSelectRule);
                    String replaceString = rules.get(numberSelectRule
                                                - 1).substring(2);
                    newIntermediateChain.append(replaceString); //
                                                    Заменяем символ
                }
            } else {
                newIntermediateChain.append(c); // Простой символ
                                                            добавляем
            }
        }

        // Обновляем интермедиатчную цепочку
        intermediateChain = newIntermediateChain.toString();
        step++;
    }

    System.out.println();
    System.out.println("Последовательность правил: " + currentRules);
    System.out.println();

    // Проверим, выполнили ли все команды
    return commandIndex == commands.length; // Если все команды были
                                                            использованы
}
```

Результат работы программы:

```
============= Задание 3 =============
========= Первая последовательность правил =========
Шаг 1
Текущая цепочка: S

Можно применить:
1: S-Ssa
2: S-b
3: S-Ab
Шаг 2
Текущая цепочка: Ssa

Можно применить:
1: S-Ssa
2: S-b
3: S-Ab

Последовательность правил: [1, 2]

Результат: false
```

```
========= Вторая последовательность правил =========
Шаг 1
Текущая цепочка: S

Можно применить:
1: S-Ssa
2: S-b
3: S-Ab
Шаг 2
Текущая цепочка: Ssa

Можно применить:
1: S-Ssa
2: S-b
3: S-Ab

Последовательность правил: [1, 2]

Результат: false
```

```
========= Третья  последовательность правил =========
Шаг 1
Текущая цепочка: S

Можно применить:
1: S-Ssa
2: S-b
3: S-Ab
Шаг 2
Текущая цепочка: Ssa

Можно применить:
1: S-Ssa
2: S-b
3: S-Ab
Шаг 3
Текущая цепочка: Absa

Можно применить:
4: A-AaA
5: A-BaB
6: A-S
Шаг 4
Текущая цепочка: AaAbsa

Можно применить:
4: A-AaA
5: A-BaB
6: A-S
```

```
Шаг 5
Текущая цепочка: Sabbsa

Можно применить:
1: S-Ssa
2: S-b
3: S-Ab
Шаг 6
Текущая цепочка: BaBabbsa

Можно применить:
7: B-b
8: B-aA
```

```
Шаг 7
Текущая цепочка: aAaSabbsa

Можно применить:
4: A-AaA
5: A-BaB
6: A-S

Можно применить:
1: S-Ssa
2: S-b
3: S-Ab

Последовательность правил: [1, 3, 4, 6, 2, 5, 8, 6, 2, 7]

Результат: false
```

```
========= Четвертая последовательность правил =========
Шаг 1
Текущая цепочка: S

Можно применить:
1: S-Ssa
2: S-b
3: S-Ab
Шаг 2
Текущая цепочка: Ssa

Можно применить:
1: S-Ssa
2: S-b
3: S-Ab

Последовательность правил: [1, 2]

Результат: false
```

Задание 4

Для каждой последовательности правил (см. варианты заданий п.2) определить, можно ли её применить при левом (правом) выводе терминальной цепочки в заданной КС-грамматике, и, если можно, построить дерево вывода.

Левый вывод:

Последовательность правил 1: 1,2,3,4,5,6,7,8,6,2,2

$\underset{1}{S} \rightarrow \underset{2}{S}sa \rightarrow bsa$

нельзя использовать данную последовательность при левом выводе, тк после применения второго правила получили терминальную цепочку.

Последовательность правил 2: 1,2,3,4,5,7,8,6,2,6,2

$\underset{1}{S} \rightarrow \underset{2}{S}sa \rightarrow bsa$

нельзя использовать данную последовательность при левом выводе, тк после применения второго правила получили терминальную цепочку.

Последовательность правил 3: 1,3,4,6,2,5,8,6,2,7,2

$\underset{1}{S} \rightarrow \underset{3}{S}sa \rightarrow \underset{4}{A}bsa \rightarrow \underset{6}{A}aAbsa \rightarrow \underset{2}{S}aAbsa \rightarrow ba\underset{5}{A}bsa \rightarrow ba\underset{8}{B}aBbsa \rightarrow baa\underset{6}{A}aBbsa \rightarrow baa\underset{2}{S}aBbsa$

$\rightarrow$

$\rightarrow baaba\underset{7}{B}bsa \rightarrow baababbsa$

нельзя применить данную последовательность

Последовательность 4: 1,2,3,4,6,6,2,2,7,8,6

$\underset{1}{S} \rightarrow \underset{2}{S}sa \rightarrow bsa$

нельзя использовать данную последовательность

Правый вывод:

Последовательность правил 1: 1,2,3,4,5,6,7,8,6,2,2

$\underset{1}{S} \rightarrow \underset{2}{S}sa \rightarrow bsa$

нельзя использовать данную последовательность при правом выводе, тк после применения второго правила получили терминальную цепочку.

Последовательность правил 2: 1,2,3,4,5,7,8,6,2,6,2

$\underset{1}{S} \rightarrow \underset{2}{S}sa \rightarrow bsa$

нельзя использовать данную последовательность при правом выводе, тк после применения второго правила получили терминальную цепочку.

Последовательность правил 3: 1,3,4,6,2,5,8,6,2,7,2

$$\underset{1}{S} \rightarrow \underset{3}{S}sa \rightarrow \underset{4}{A}bsa \rightarrow \mathrm{Aa}\underset{6}{A}bsa \rightarrow Aa\underset{2}{S}bsa \rightarrow \underset{5}{A}abbsa \rightarrow Ba\underset{8}{B}abbsa \rightarrow Baa\underset{6}{A}abbsa \rightarrow Baa\underset{2}{S}abbsa$$

$$\rightarrow$$

$$\rightarrow \underset{7}{B}aababbsa \rightarrow baababbsa$$

нельзя использовать данную последовательность правил

Последовательность 4: 1,2,3,4,6,6,2,2,7,8,6

$\underset{1}{S} \rightarrow \underset{2}{S}sa \rightarrow bsa$

нельзя использовать данную последовательность

Задание 5. Написать программу, определяющую, можно ли применить заданную последовательность правил при выводе цепочки в заданной КС-грамматике

Код программы:

```
  public static boolean output(ArrayList<String> rules, int[] commands) {
        String startChain = "S"; // Начальная цепочка
        String intermediateChain = startChain; // Промежуточная цепочка
        ArrayList<Integer> currentRules = new ArrayList<>(); // Список
                                                    применённых правил
        int commandIndex = 0; // Индекс для отслеживания текущей команды

        // Цикл продолжается, пока есть большие буквы в промежуточной
                                                                цепочке
        while (!intermediateChain.equals(intermediateChain.toLowerCase()))
{
            StringBuilder newIntermediateChain = new StringBuilder(); //
                            Новый объект для хранения изменённой цепочки
            ArrayList<Integer> applicableRules = new ArrayList<>(); //
                                                Список применимых правил

            // Перебираем каждый символ в промежуточной цепочке
            for (char c : intermediateChain.toCharArray()) {
                // Проверяем, является ли символ заглавной буквой
                if (Character.isUpperCase(c)) {
                    // Ищем применимые правила
                    for (int j = 0; j < rules.size(); j++) {
                        String currentRule = rules.get(j);
                        if (currentRule.charAt(0) == c) {
                            applicableRules.add(j + 1); // Сохраняем
                                            индекс правила (нумерация с 1)
                        }
                    }

                    // Если нет применимых правил
                    if (applicableRules.isEmpty()) {
                        newIntermediateChain.append(c); // Оставляем
                                                    символ без изменений
                    } else {
                        // Если есть доступные команды, используем их
                        int numberSelectRule;
                        if (commandIndex < commands.length) {
                            numberSelectRule = commands[commandIndex++];
```

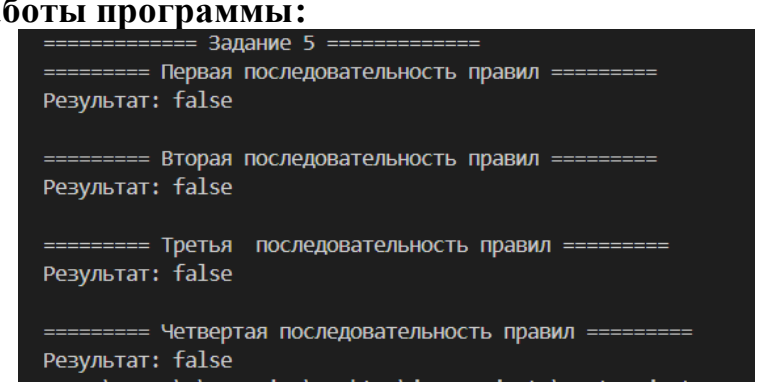
```
                        // Получаем номер правила из массива команд
                    } else {
                        newIntermediateChain.append(c); // Если команд
                                нет, оставляем символ без изменений
                        continue; // Переходим к следующему символу
                    }

                    // Проверяем, что выбранное правило применимо
                    if (numberSelectRule > 0 && numberSelectRule <=
                                                    rules.size()) {
                        String replaceString =
                         rules.get(numberSelectRule - 1).substring(2);
                                                // Получаем строку замены
                        newIntermediateChain.append(replaceString); //
                                                    Заменяем символ
                        currentRules.add(numberSelectRule); //
                                                Отслеживаем применённое правило
                    } else {
                        newIntermediateChain.append(c); // Если номер
                          неправильный, оставляем символ без изменений
                    }
                }
            } else {
                newIntermediateChain.append(c); // Простые (не
                            заглавные) символы добавляем без изменений
            }
        }

        // Обновляем промежуточную цепочку на основе новых значений
        intermediateChain = newIntermediateChain.toString();
    }

    // Проверяем, были ли использованы все команды
    return commandIndex == commands.length; // Возвращаем true, если
                                        все команды были использованы
}
```

Результат работы программы:



Задание 6

Для каждой последовательности правил определить, можно ли ее применить при выводе терминальной цепочки в заданной КС-грамматике, и, если можно, построить дерево вывода и записать эквивалентные левый и правый.

Последовательность правил 1: 1,2,3,4,5,6,7,8,6,2,2

$\underset{1}{S} \to \underset{2}{S}sa \to bsa$

нельзя использовать данную последовательность, т.к после применения второго правила получили терминальную цепочку.

Последовательность правил 2: 1,2,3,4,5,7,8,6,2,6,2

$\underset{1}{S} \to \underset{2}{S}sa \to bsa$

нельзя использовать данную последовательность, т.к после применения второго правила получили терминальную цепочку.

Последовательность правил 3: 1,3,4,6,2,5,8,6,2,7,2

$$\underset{1}{S} \to \underset{3}{S}sa \to \underset{4}{A}bsa \to \underset{6}{\text{Aa}}Absa \to \underset{2}{S}aAbsa \to ba\underset{5}{A}bsa \to baB\underset{8}{a}Babbsa \to baBaa\underset{6}{A}abbsa$$
$$\to baBaa\underset{2}{S}abbsa \to ba\underset{7}{B}aabbsa \to babaabbsa$$

Нельзя использовать данную последовательность, т.к после выполнения 7 правила получили терминальную цепочку, следовательно последнее второе правило не выполнится.

Последовательность 4: 1,2,3,4,6,6,2,2,7,8,6

$\underset{1}{S} \to \underset{2}{S}sa \to bsa$

нельзя использовать данную последовательность

Вывод: В ходе выполнения лабораторной работы изучили основные понятия теории формальных языков и грамматик.