Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

**Лабораторная работа №1.**

**«Модели Машины Тьюринга и Алгоритмов Маркова»**

Выполнил студент гр. РИС-24-2б

Долганов Даниил Вячеславович

Проверил:

Доц. Каф. ИТАС

Ольга Андреевна Полякова

(оценка) (подпись)

(дата)

г. Пермь, 2024

**Нормальные алгоритмы Маркова**

Модель алгоритмов Маркова преобразует исходные слова, с помощью заданных правил, которые могут заменять часть исходной строки, кроме правил в алгоритмах Маркова используются приоритеты их применения.

Слово - срока из символов.

Работа с алгоритмами:

1) Правило находит первое свое вхождение в исходной строке и заменяет его на правую часть ( указанные данные ).

2) Правила выполняются по порядку их номеров.

3) Если правило не применятся к исходной строке, то алгоритм переходит к проверке следующего по порядку правила.

4) Если какое-то правило выполнено, то алгоритм возвращается к проверке правил, начиная с самого начала.

5) Алгоритм завершается, если ни одно правило не применимо

Примечания для работы с подстановками:

Виды постановок:

1. Нетерминальные - подстановки, после выполнения которых, выполнение программы продолжается.

2. Терминальные - говорят, о принудительном завершении программы, после выполнения замены.

3. В алгоритмах Маркова терминальные подстановка оканчиваются точкой.

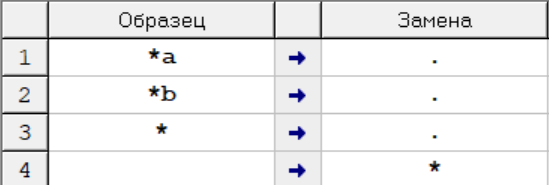
4. Если правая часть правила пустая, то левая часть заменятся на пустую строку.

5. Если левая часть пустая, то в начало слова добавляется правая часть этого правила.

***Задача № 1:*** Из непустого слова P, состоящего из символов a и b, пустое слово не менять

Допустим исходное слово - bbaba.

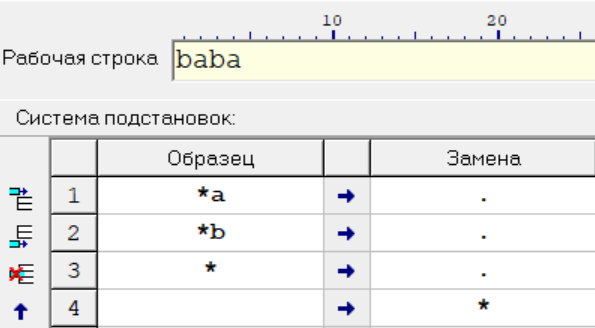
Для решения данной зачади, необходимо пометить первый символ. Чтобы это сделать можно воспользоваться тем фактом, что если левая часть правила пустая, то символ из правой части добавляется в начало слова. Из этого следует, что правило ( void ->\* ) добавит \* в начало слова. Составим правила алгоритма и выставим их приоритеты:



Алгоритм будет состоять из трех терминальных правил, и одного нетерминального. Первые два заменят первый символ числа и завершат программу, третье не даст программе зациклится если введена пустая строка, а четвертое пометит первый символ.

Для исходного слова bbaba:

1. bbaba -> \*bbaba // 4-ое правило
2. \*bbaba |-> baba // 1-ое правило

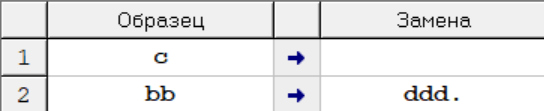


**Вывод: Алгоритм сработал. Задача решена.**

***Задача № 2:*** В слове P, состоящем из символов a,b,c,d требуется удалить все вхождения символа c, а после заменить первое вхождение подслова bb на ddd.

Пусть исходное слово – abbcabbca.

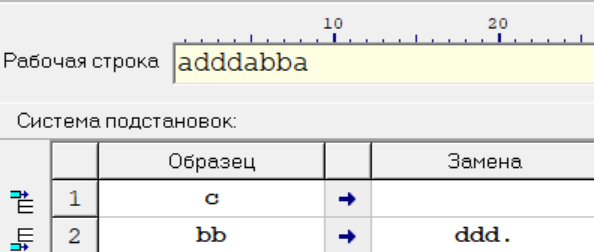
Составим правила алгоритма и выстроим их приоритеты:



Алгоритм будет состоять из одной нетерминального правила и одного терминального. Первое правило заменить все символы с в слове. Второе правило заменит первое вхождение подслова bb на ddd и завершит выполнение программы.

Из исходного слова abbcabbca:

1. abbcabbca -> abbabbca // 1-ое правило
2. abbabbca -> abbabba // 1-ое правило
3. abbabba |-> adddabba // 2-ое правило



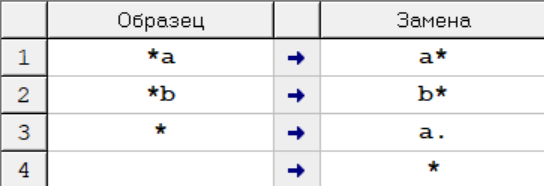
**Вывод: Алгоритм сработал. Задача решена.**

***Задача № 3:*** Необходимо приписать символ а к концу слова P, состоящему из символов а и b.

Пусть исходное слово – bbab.

Для выполнение этой задачи, на подобие задачи №1, необходимо пометить конец числа. Для этого добавим в начало \*, протянем ее в конец числа и заменим на символ а.

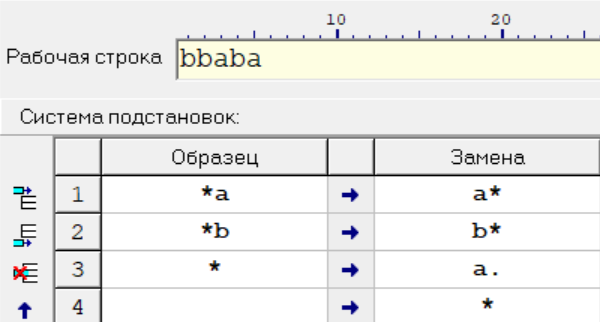
Составим правила алгоритма и выставим их приоритеты:



В алгоритме используются 3 нетерминальных и одно терминальное правило. Первые 2 протаскивают \* в конец слова, 3-тье заменяет \* на символ а и завершает выполнение программы, 4-ое добавляет \* в начало слова.

Для слова bbab:

1. bbab -> \*bbab // 4-ое правило
2. \*bbab -> b\*bab // 2-ое правило
3. b\*bab -> bb\*ab // 2-ое правило
4. bb\*ab -> bba\*b // 1-ое правило
5. bba\*b -> bbab\* // 2-ое правило
6. bbab\* |-> bbaba // 3-е правило



**Вывод: Алгоритм сработал. Задача решена.**